ESCUELA POLITECNICA NACIONAL

FACULTAD DE INGENIERIA ELECTRICA

SISTEMA DIDACTICO DE DESARROLLO BASADO EN EL MICROCONTROLADOR M68HC11

TESIS PREVIA A LA OBTENCION DEL TITULO DE:

INGENIERO EN ELECTRONICA Y CONTROL

ANGEL VINICIO ACOSTA VILLACIS

INGENIERO EN ELECTRONICA Y

TELECOMUNICACIONES

WILSON WILFRIDO PASTUISACA ORELLANA

Quito, Junio, 1998

LISTADO DE PROGRAMAS DE LAS PRACTICAS IMPLEMENTADAS

```
**********************************
*PROGRAMA PRACTICA 1 NUMERAL 4.1.1*
**********************************

PIOC    EQU $1002   *REGISTRO DE CONTROL DE I/O DEL PORTICO C*
PORTC   EQU $1003   *REGISTRO DE I/O DEL PORTICO C*
DDRC    EQU $1007   *REGISTRO DE DIRECCION DE DATOS DEL PORTICO C*

        ORG $C000
        LDAA #$03   *INSTRUCCIONES PARA INICIALIZAR*
        STAA PIOC   *EL PORTICO C COMO PORTICO DE*
        LDAA #$FF   *SALIDA*
        STAA DDRC

INICIO  LDAA #$01   *CARGAR NIVEL LOGICO PARA EL ENCENDIDO DE LEDS*
CLEARC  CLC
        STAA PORTC
        JSR RETARDO *RETARDO PARA VISUALIZAR EL ENCENDIDO SECUENCIAL*
        ROLA
        CMPA #$80
        BNE CLEARC  *SALTO CUANDO TERMINE DE ENCENDER TODOS LOS LEDS*
        STAA PORTC
        JSR RETARDO
        JMP INICIO

*******************************
*SUBRUTINA DE RETARDO DE 16 ms*
*******************************
RETARDO LDX #$7FFF
DECREM  DEX
        BNE DECREM
        RTS

*****
*END*
*****
```

```
**********************************
*PROGRAMA PRACTICA 1 NUMERAL 4.1.2*
**********************************

PIOC    EQU $1002   *REGISTRO DE CONTROL DE I/O DEL PORTICO C*
PORTC   EQU $1003   *REGISTRO DE I/O DEL PORTICO C*
DDRC    EQU $1007   *REGISTRO DE DIRECCION DE DATOS DEL PORTICO C*
DIPSW   EQU $6800   *DIRECCION DE ENTRADAS DIGITALES*
        ORG $C000
        LDAA #$03   *INSTRUCCIONES PARA INICIALIZAR*
        STAA PIOC   *EL PORTICO C COMO PORTICO DE*
        LDAA #$FF   *SALIDA*
        STAA DDRC


INICIO  LDAA DIPSW  *CARGA EL DATO DE LOS DIP-SWITCHS*
        STAA PORTC  *ENVIA EL DATO A LOS LEDS*
        JMP INICIO

*****
*END*
*****
```

```
**********************************
*PROGRAMA PRACTICA 1 NUMERAL 4.2.1*
**********************************


************************
*ASIGNACION DE ETIQUETAS*
************************
LCD0    EQU #$6000      *DIRECCIONES DEL LCD*
LCD1    EQU #$6001
LCD2    EQU #$6002
LCD3    EQU #$6003
INTERM  EQU #$C100      *DIRECCION DE ALMACENAMIENTO INTERMEDIO*

        ORG $C000
**************************************
*ENCERAR ACC, DISPLAY Y MEMORIA INTERM*
**************************************
        CLR LCD0        *ENCERARAMIENTO DE LOCALIDADES*
        CLR LCD1        *DE MEMORIA Y ACUMULADOR*
        CLR LCD2
        CLR LCD3
        CLR INTERM
        CLRA
*******************
*PROGRAMA PRINCIPAL*
*******************
SUMAR1  LDX #LCD0       *CARGAR LOS PUNTEROS DE DIRECCIONES *
        LDY #LCD1       *CON LCD0 Y LCD1*
SUMAR   ADDA #$01       *INSTRUCCIONES PARA IMPLEMENTAR*
        DAA             *EL CONTADOR ASCENDENTE*
        JSR MOSTRAR     *SACAR LOS DATOS AL DISPLAY*
        JSR RETARDO     *RETARDO PARA VISUALIZAR LOS NUMEROS EN EL LCD*
        CMPA #$00
        BNE SUMAR
        LDAA INTERM
        LDX #LCD2       *CARGAR LOS PUNTEROS DE DIRECCIONES*
        LDY #LCD3       *CON LCD2 Y LCD3*
        ADDA #$01
        DAA
        JSR MOSTRAR
        STAA INTERM
        JSR RETARDO
        JMP SUMAR1

**********************
*SUBRUTINA DE RETARDO*
**********************
RETARDO PSHX        *SALVAR LOS DATOS DE LOS REGISTROS*
        PSHY
        LDX #$7FFF
DECX    DEX
        BNE DECX
        LDY #$7FFF
DECY    DEY
        BNE DECY
        PULY        *REESTABLECER LOS DATOS DE LOS REGISTROS*
        PULX
        RTS
```

```
*****************************************
*SUBRUTINA PARA CARGAR LOS DATOS A DISPLAY*
*****************************************
MOSTRAR RORA      *INSTRUCCIONES DE ROTACION PARA DIVIDIR EL*
        RORA      *DATO EN DOS DIGITOS Y SACARLOS AL DISPLAY*
        RORA
        RORA
        STAA 0,Y
        ROLA
        ROLA
        ROLA
        ROLA
        STAA 0,X
        RTS

*****
*END*
*****
```

```
**********************************
*PROGRAMA PRACTICA 1 NUMERAL 4.2.2*
**********************************

LCD0    EQU #$6000  *DIRECCIONES DEL LCD*
LCD1    EQU #$6001
LCD2    EQU #$6002
LCD3    EQU #$6003
DIPSW   EQU #$6800  *DIRECCION DE ENTRADAS DIGITALES*

        ORG $C000
********************
*PROGRAMA PRINCIPAL*
********************
        CLR LCD0        *ENCERAMIENTO DE LOCALIDADES DEL LCD*
        CLR LCD1
        CLR LCD2
        CLR LCD3
        LDY #$112C      *CARGAR EL VALOR DEL RETARDO EXTERNO*
RETARH  DEY
        JSR RETARL      *SUBRUTINA DE RETARDO INTERNO*
        BNE RETARH
        LDAA DIPSW      *INSTRUCCIONES PARA SACAR AL*
        STAA LCD0       *DISPLAY EL VALOR DE LOS*
        RORA            *DIP-SWITCHS*
        RORA
        RORA
        RORA
        STAA LCD1
        ROLA
        ROLA
        ROLA
        ROLA
FIN     JMP FIN

*****************************************
*SUBRUTINA PARA GENERAR EL RETARDO INTERNO*
*****************************************
RETARL  PSHY            *EL RETARDO GENERADO EN ESTA SUBRUTINA*
        TPA             *MULTIPLICADO POR EL VALOR DE RETARDO*
        LDY #$00FF      *EXTERNO NOS DA EXACTAMENTE CUATRO*
DECY1   DEY             *SEGUNDOS*
        BNE DECY1
        TAP
        PULY
        RTS

*****
*END*
*****
```

```
*********************************
*PROGRAMA PRACTICA 2 NUMERAL 4.1*
*********************************

REGBS   EQU $1000      *DIRECCION DE INICIO DE LOS REGISTROS DEL MCU*
OPTION  EQU REGBS+$39  *REGISTRO QUE DEFINE LAS OPCIONES*
*                      *DE CONFIGURACION DEL SISTEMA*
JSCI    EQU $00C4      *VECTORES DE SALTO A INTERRUPCIONES*
JCLM    EQU $00FD      *POR SCI Y CLM*
LCD0    EQU #$6000     *DIRECCIONES DE LOS DISPLAYS*
LCD1    EQU #$6001
LCD2    EQU #$6002
LCD3    EQU #$6003
DIPSW   EQU #$6800     *DIRECCION DE LOS DIPSWITCHS*

        ORG $C000

************************************************
* PROGRAMA PARA HABILITAR LA MEMORIA EEPROM INTERNA*
************************************************

        LDAA #$93      *VALOR QUE DEFINE LA CONFIGURACION DEL*
        STAA OPTION    *SISTEMA LUEGO DE UN RESET*
        JSR VECINIT

*********************************************************************
*PROGRAMA PRINCIPAL MUESTRA EN DISPLAY EL VALOR DE LOS DIP-SWITCHS*
*********************************************************************
        CLR  LCD0       *ENCERAR LOCALIDADES DE DISPLAY*
        CLR  LCD1
        CLR  LCD2
        CLR  LCD3
CARGAR  LDAA DIPSW
        STAA LCD0
        LSRA
        LSRA
        LSRA
        LSRA
        STAA LCD1
        LSLA
        LSLA
        LSLA
        LSLA
        JMP CARGAR

*************************************************************
*SUBRUTINA VECINIT                                          *
*REALIZA UN MONITOREO DE TODOS LOS ESTADOS DE LOS VECTORES*
*DE INTERRUPCION Y PONE AL SISTEMA EN EL MODO DE STOP SI   *
*EXISTEN INTERRUPCIONES RELACIONADAS CON VECTORES NO       *
*INICIALIZADOS                                             *
*************************************************************

VECINIT LDY #JSCI        *PUNTERO DE LOS VECTORES EN LA RAM*
        LDX #STOPIT      *PUNTERO DE LA SUBRUTINA STOPIT*
        LDD #$7E03
VECLOOP CMPA 0,Y
        BEQ VECNEXT
        STAA 0,Y
        STX 1,Y
VECNEXT ABY             *A¥ADE 3 LOCALIDADES AL PUNTERO DE VECTORES*
```

```
        CPY #JCLM+3
        BNE VECLOOP
        RTS
STOPIT  LDAA #$50       *HABILITA EL MODO STOP,IRQ Y*
        TAP             *DESHABILITA XIRQ*
        STOP
        JMP STOPIT

*****
*END*
*****
```

```
*********************************
*********************************
TMSK1    EQU #$1022 *DIRECCION DEL REGISTRO DE MASCARAS PARA IC1*
TFLG1    EQU #$1023 *DIRECCION DEL REGISTRO DE BANDERAS DE IC1*
TCTL2    EQU #$1021 *DIRECCION DEL REGISTRO EN LA QUE SE DEFINE LA
*                   *ACCION QUE GENERA LA INTERRUPCION INTC1*
PORTC    EQU $1003       *DIRECION DEL PORTICO C*
LCD0     EQU #$6000      *DIRECCIONES DEL LCD*
LCD1     EQU #$6001
LCD2     EQU #$6002
LCD3     EQU #$6003
INTERM   EQU #$C100      *DIRECCION DE LOCALIDAD INTERNA*
DIPSW    EQU #$6800      *DIRECCION DE DIP-SWITCHS*
TECLAS   EQU #$6400      *DIRECCION DE TECLAS*

         ORG $00E8
         JMP INTIC1  *SALTO A SUBRUTINA DE INTERRUPCION IC1*

         ORG $C000
         LDAA #$03       *INICIALIZACION DEL PORTICO C*
         STAA $1002
         LDAA #$FF
         STAA $1007
         LDAA #$20
         STAA TCTL2  *PARA QUE INTERRUPCION INTIC1 SE GENERE POR FLANCO
NEGATIVO*
         LDAA #$04
         STAA TMSK1  *QUITAR MASCARA A IC1*
         LDAA #$88
         TAP         *HABILITAR INTERRUPCIONES*

*****************
*PROGRAMA PRINCIPAL*
*****************
PROGPRI JMP PROGPRI

************************************************
*SUBRUTINA DE ATENCION A INTERRUPCION EXTERNA IC1*
*PARA DETERMINAR LA TECLA PRESIONADA*
************************************************
INTIC1   CLR LCD0       *ENCERAR DISPLAY*
         CLR LCD1
         CLR LCD2
         CLR LCD3
         LDAA TECLAS    *PROGRAMA PARA DETERMINAR CUAL FUE LA TECLA
PRESIONADA*
         CMPA #$01
         BNE TECLA2
         JSR PROG1
         JMP SALIR
TECLA2   CMPA #$02
         BNE TECLA3
         JSR PROG2
         JMP SALIR
TECLA3   CMPA #$04
         BNE TECLA4
         JSR PROG3
         JMP SALIR
TECLA4   CMPA #$08
         BNE SALIR
         JSR PROG4
```

```
SALIR   LDAA #$04
        STAA TFLG1    *LIMPIAR BANDERA DE IC1*
        LDAA #$88
        TAP           *HABILITAR INTERRUPCIONES ANTES DE SALIR*
        RTI

*****************************************
*SUBRUTINA DE ATENCION A TECLA1 LA CUAL   *
*PERMITE ENCENDER SECUENCIALMENTE LOS LEDS*
*****************************************
PROG1   PSHA
        LDAA #$01       *CARGAR NIVEL LOGICO PARA ENCENDER LEDS*
CLEARC  CLC
        STAA PORTC      *ENVIO DE DATOS A LOS LEDS*
        JSR RETARDO     *RETARDO DE PRESENTACION PARA LOS DATOS*
        ROLA
        CMPA #$80
        BNE CLEARC
        STAA PORTC
        JSR RETARDO
        PULA
        RTS

RETARDO LDX #$3FFF
DECRE1  DEX
        BNE DECRE1
        RTS

*********************************************
*SUBRUTINA DE ATENCION A TECLA2 LA CUAL      *
*ENVIA LOS DATOS DE LOS DIP-SWITCHS A LOS LEDS*
*********************************************
PROG2   PSHA
        LDY #$0FFFF
DECY    LDAA DIPSW      *CARGAR LOS DATOS DE LOS DIP SWITCHS*
        STAA PORTC      *ENVIO LOS DATOS A LOS LEDS*
        DEY
        BNE DECY
        PULA
        RTS

*********************************************
*SUBRUTINA DE ATENCION A TECLA3 LA CUAL       *
*CUENTA INDEFINIDAMENTE Y ENVIA LOS DATOS AL LCD*
*********************************************
PROG3   CLR LCD0        *ENCERAMIENTO DEL LCD, LOCALIDADES INTERNAS*
        CLR LCD1        *Y ACUMULADOR*
        CLR LCD2
        CLR LCD3
        CLR INTERM
        CLRA
SUMAR1  LDX #LCD0       *PUNTEROS PARA EL LCD*
        LDY #LCD1
SUMAR   ADDA #$01
        DAA
        JSR MOSTRAR
        JSR RETARD1
        CMPA #$00
        BNE SUMAR
        LDAA INTERM
        LDX #LCD2
```

```
         LDY #LCD3
         ADDA #$01
         DAA
         JSR MOSTRAR      *SUBRUTINA PARA MOSTRAR LOS DIGITOS EN EL LCD*
         STAA INTERM
         JSR RETARD1
         JMP SUMAR1
RETARD1  PSHX
         PSHY
         LDX #$7FFF
DECX     DEX
         BNE DECX
         LDY #$7FFF
DECY2    DEY
         BNE DECY2
         PULY
         PULX
         RTS
MOSTRAR  RORA
         RORA
         RORA
         RORA
         STAA 0,Y
         ROLA
         ROLA
         ROLA
         ROLA
         STAA 0,X
         RTS


*********************************************
*SUBRUTINA DE ATENCION A TECLA4 LA CUAL      *
*GENERA RETARDO DE 4 SEGUNDOS Y ENVIA EL DATO*
*DE LOS DIP-SWITCHS AL LCD                   *
*********************************************
PROG4    PSHA
         PSHY
         LDY #$112C       *FACTOR PARA EL RETARDO INTERNO*
RETARH   DEY
         JSR RETARL
         BNE RETARH
         LDAA DIPSW       *CARGO EL DATO DEL DIP-SWITCHS
         STAA LCD0
         RORA
         RORA
         RORA
         RORA
         STAA LCD1
         ROLA
         ROLA
         ROLA
         ROLA
         PULY
         PULA
         RTS

RETARL   PSHY
         PSHA
         TPA
         LDY #$00FF       *FACTOR DE RETARDO INTERNO*
```

```
DECY1   DEY
        BNE DECY1
        TAP
        PULA
        PULY
        RTS

*****
*END*
*****
```

****

```
*PROGRAMA PRACTICA 3 NUMERAL 4.1*
*********************************

TCNT     EQU #$100E *DIRECCION DEL REGISTRO DEL TIMER/COUNTER*
TOC2     EQU #$1018 *DIRECCION DEL REGISTRO DEL OUTPUT COMPARE 2*
TMSK1    EQU #$1022 *DIRECCION DEL REGISTRO DE MASCARAS PARA OC2*
TFLG1    EQU #$1023 *DIRECCION DEL REGISTRO DE BANDERAS DE OC2*
TCTL2    EQU #$1021 *DIRECCION DEL REGISTRO EN LA QUE SE DEFINE LA  ACCION
QUE GENERA LA INTERRUPCION INTC3 (FLANCO O ESTADO)*
LCD0     EQU #$6000
LCD1     EQU #$6001
LCD2     EQU #$6002
LCD3     EQU #$6003
LCD4     EQU #$6004
LOCINT   EQU #$D000 *LOCALIDADES DE RAM PARA GUARDAR EL VALOR DEL CONTADOR*
LOCINT1  EQU #$D001
INCR12   EQU #$D002 *LOCALIDAD DE RAM PARA GUARDAR EL NUMERO DE VECES QUE SE
*                    INGRESA A LA INTERRUPCION INTIRQ PARA COMPLETAR LA BASE
DE TIEMPO*

         ORG $00E2
         JMP INTIC3   *SALTO A SUBRUTINA DE INTERRUPCION INPUT COMPARE 3*

         ORG $00DC
         JMP INTOC2   *SALTO A SUBRUTINA DE INTERRUPCION OUTPUT COMPARE 2*

         ORG $C000

*************************************************
*INICIALIZACION DE REGISTROS USADOS EN EL PROGRAMA*
*************************************************
         LDD TCNT
         ADDD #$F424   * GENERAR RETARDO DE 30 ms*
         STD TOC2
         LDAA #$02
         STAA TCTL2   *PARA QUE INTERRUPCION INTIC3 SE GENERE POR FLANCO
NEGATIVO*
         LDAA #$41
         STAA TMSK1   *QUITAR MASCARA A IC3 E OC2*
         LDAA #$88
         TAP          *HABILITAR INTERRUPCIONES*

*******************
*ENCERAR REGISTROS*
*******************
         CLRA
         CLRB
         CLR LOCINT
         CLR LOCINT1
         CLR INCR12
         CLR LCD0
         CLR LCD1
         CLR LCD2
         CLR LCD3

********************
*PROGRAMA PRINCIPAL*
********************
PROGPRI JMP PROGPRI

*************************************************
```

```
*SUBRUTINA DE INTERRUPCION EXTERNA (INPUT COMPARE 3)*
*************************************************
INTIC3  LDAA #01
        ADDA LOCINT
        DAA
        STAA LOCINT
        BCC FINAL
        LDAB #01
        ADDB LOCINT1
        STAB LOCINT1
FINAL   LDAA #$01
        STAA TFLG1    *LIMPIAR BANDERA DE IC1*
        LDAA #$88
        TAP           *HABILITAR INTERRUPCIONES ANTES DE SALIR*
        RTI

*************************************************
*SUBRUTIMA DE INTERRUPCION  TOC2 (BASE DE TIEMPO)*
*************************************************
INTOC2  LDAA #$98
        TAP            *DESHABILITO INTERRUPCIONES*
        INC INCR12
        LDAB INCR12
        CMPB #12
        BNE CARGAR
        LDX #LCD1
        LDY #LCD2
        LDAA LOCINT
        JSR MOSTRAR
        LDX #LCD3
        LDY #LCD4
        LDAA LOCINT1
        JSR MOSTRAR
        CLR LOCINT
        CLR LOCINT1
        CLRB
CARGAR  STAB INCR12
        LDD TCNT
        ADDD #$F424
        STD TOC2       *CARGAR EL VALOR DEL RETARDO*
        LDAA #$40
        STAA TFLG1     *LIMPIAR BANDERA DE OC2*
        LDAA #$88
        TAP            *HABILITO INTERRUPCIONES*
        RTI

*************************************************************
*SUBRUTINA PARA SACAR EL VALOR DE LA VELOCIDAD AL DISPLAY*
*************************************************************
MOSTRAR RORA
        RORA
        RORA
        RORA
        STAA 0,Y
        ROLA
        ROLA
        ROLA
        ROLA
        STAA 0,X
        RTS
```

```
*********************************
*PROGRAMA PRACTICA 4 NUMERAL 4.1*
*********************************
SCCR1   EQU #$102C *DIRECCION DEL REGISTRO QUE DEFINE EL FORMATO DE
TRANSMISION SCI*
SCCR2   EQU #$102D *DIRECCION DEL REGISTRO QUE HABILITA INTERRUPCIONES POR
SCI*
SCSR    EQU #$102E *DIRECCION DEL REISTRO QUE INDICA EL ESTADO DE SCI*
BAUD    EQU #$102B *DIRECCION DEL REGISTRO QUE DEFINE LA VELOCIDAD DEL SCI*
SCDR    EQU #$102F *DIRECCION DEL REGISTRO QUE ALMACENA EL DATO A SER
TRANSMITIDO O RECIBIDO*

TCNT    EQU #$100E *DIRECCION DEL REGISTRO DEL TIMER/COUNTER*
TOC2    EQU #$1018 *DIRECCION DEL REGISTRO DEL OUTPUT COMPARE 2*
TMSK1   EQU #$1022 *DIRECCION DEL REGISTRO DE MASCARAS PARA OC2*
TFLG1   EQU #$1023 *DIRECCION DEL REGISTRO DE BANDERAS DE OC2*
TCTL2   EQU #$1021 *DIRECCION DEL REGISTRO EN LA QUE SE DEFINE LA*
*                  GENERA LA INTERRUPCION INTC3 (FLANCO O ESTADO)*
LCD0    EQU #$6000
LCD1    EQU #$6001
LCD2    EQU #$6002
LCD3    EQU #$6003
UNIDAD  EQU $00
DECENA  EQU $01
CENTENA EQU $03
UNIMIL  EQU $04
LOCINT  EQU #$D000 *LOCALIDADES DE RAM PARA GUARDAR EL VALOR DEL CONTADOR*
LOCINT1 EQU #$D001
INCR12  EQU #$D002 *LOCALIDAD DE RAM PARA GUARDAR EL NUMERO DE VECES QUE SE
*                  INGRESA A LA INTERRUPCION INTIRQ PARA COMPLETAR LA BASE
DE TIEMPO*

        ORG $00E2
        JMP INTIC3  *SALTO A SUBRUTINA DE INTERRUPCION INPUT COMPARE 3*

        ORG $00DC
        JMP INTOC2  *SALTO A SUBRUTINA DE INTERRUPCION OUTPUT COMPARE 2*
        ORG $C000

************************************************
*INICIALIZACION DE REGISTROS USADOS EN EL PROGRAMA*
************************************************

        LDAA #$08
        STAA SCCR1 *COMUNICACION 1-8-1*
        LDAA #$30
        STAA BAUD  *SETEA VELOCIDAD A 9600 BPS*
        LDAA #$01
        STAA $4000 *HABILITAR FLIP-FLOP*
HABIL   LDAA #$04
        STAA SCCR2 *HABILITAR RECEPCION*
        LDX #SCSR
ESPE    BRCLR 0,X #$20 ESPE *ESPERA POR RECEPCION DE DATO COMPLETO*
        LDAA SCSR
        LDAA SCDR
        CMPA #$01
        BNE HABIL
        LDD TCNT
        ADDD #$F424  * GENERAR RETARDO DE 30 ms*
        STD TOC2
        LDAA #$02
```

```
        STAA CENTENA
        TBA
        ANDA #$0F
        STAA 0,Y
        STAA DECENA
        LDX #LCD0
        LDY #LCD3
        LDAA LOCINT1
        TAB
        ANDA #$F0
        RORA
        RORA
        RORA
        RORA
        STAA 0,X
        STAA UNIDAD
        TBA
        ANDA #$0F
        STAA 0,Y
        STAA UNIMIL
        JSR ORDENAR
        CLR LOCINT
        CLR LOCINT1
        CLRB
CARGAR  STAB INCR12
        LDD TCNT
        ADDD #$F424
        STD TOC2        *CARGAR EL VALOR DEL RETARDO*
        LDAA #$40
        STAA TFLG1      *LIMPIAR BANDERA DE OC2*
        LDAA #$88
        TAP             *HABILITO INTERRUPCIONES*
        RTI

*********************************************
*SUBRUTINA PARA ORDENAR VALORES A TRANSMITIR*
*********************************************
ORDENAR
        PSHA
        LDAA UNIMIL
        JSR TRANSMI
        LDAA CENTENA
        JSR TRANSMI
        LDAA DECENA
        JSR TRANSMI
        LDAA UNIDAD
        JSR TRANSMI
        PULA
        RTS

*********************************************
*SUBRUTINA PARA TRANSMITIR DATOS AL COMPUTADOR*
*********************************************
TRANSMI PSHX
        PSHY
        PSHA
        PSHB
        LDX #SCSR
        LDY #SCCR2
        BSET 0,Y #$08
        STAA SCDR
```

```
ESPERA  BRCLR 0,X #$40 ESPERA  *ESPERA MIENTRAS NO SE HAYA TERMINADO LA
TRANSMISION DEL DATO ANTERIOR*
        BCLR 0,Y #$08
        LDAB SCSR   *INSTRUCIONES PARA LIMPIAR LA BANDERA DE  FIN DE
TRANSMISION*
        STAB SCDR
        PULB
        PULA
        PULY
        PULX
        RTS
```

```
*************************************************
*PROGRAMA PRACTICA 5 MANEJO DE DISPLAY ALFANUMERICO*
*************************************************
************************************
*         INICIALIZACION DEL PORTICO B*
************************************
DIGIT   EQU $25          *LOCALIDAD PARA ALMACENAR DIGITOS DE 0-9*
CURSOR  EQU $20          *LOCALIDAD PARA ALMACENAR LA POSICION DEL CURSOR*
EOT     EQU $04
RAMI    EQU 30
PORTB   EQU #$1004       *DIRECCION DEL PORTICO B*
TCTL2   EQU #$1021       *DIRECCION DEL REGISTRO TCTL2*
TMSK1   EQU #$1022       *DIRECCION DEL REGISTRO TMSK1*
TFLG1   EQU #$1023       *DIRECCION DEL REGISTRO TFLF1*
DATO    EQU $22          *DIRECCION DEL DATO*
TECLAS  EQU #$6400       *DIRECCION MEN EXT DE TECLAS*

        ORG $00E8        *DIRECCION DEL SALTO DE LA INTERRUPCION IC1*
        JMP INTIC1
        ORG $C000        *DIRECCION DE INICIO DE PROGRAMA*
        LDAA #$88        *HABILITACION DE INTERRUPCIONES*
        TAP
        LDAA #$20        *HABILITACION DE IC1 POR FLANCO DE BAJADA*
        STAA TCTL2
        LDAA #$04        *HABILITACION DE LA INTERRUPCION IC1*
        STAA TMSK1
        JSR INICDIS      *LLAMADO A SUBRUTINA DE INICIALIZACION DISPLAY*
        JSR BLINK0       *LLAMADO A SUBRUTINA PARA BORRAR BLINK0 Y SHIFT*
        JSR SEND         *ENVIO DE MENSAJE "V&W-HC11 OK"
        JSR BLINK0
        JSR CLEARD
        LDX #TEXTO
        JSR SENDM
        LDAA #$00        *INICIALIZAMOS DATO CON 00*
        STAA DATO
        JSR PRESENT      *LLAMADO A SUBRUTINA PARA PRESENTAR 000*

********************
*PROGRAMA PRINCIPAL*
********************

PROGPRI JMP PROGPRI

*****************************************************
*SUBRUTINA DE ATENCION A INTERRUPCION EXTERNA INTIC1*
*****************************************************
INTIC1  LDAA #$98        *DESHABILITO INTERRUPCIONES
        TAP
        LDAA TECLAS      *LECTURA DE TECLA PRESIONADA*
        CMPA #$01
        BNE TECLA2
        JSR PROG1        *LLAMADO A SUBRUTINA DE INCREMENTO DEL DATO EN 1*
        JMP SALIR
TECLA2  CMPA #$02
        BNE TECLA3
        JSR PROG2        *LLAMADO A SUBRUTINA DE DECREMENTO DEL DATO EN 1*
        JMP SALIR
TECLA3  CMPA #$04
        BNE TECLA4
```

```
        JSR PROG3       *LLAMADO A SUBRUTINA DE INCREMENTO DEL DATO EN 5*
        JMP SALIR
TECLA4  CMPA #$08
        BNE SALIR
        JSR PROG4       *LLAMADO A SUBRUTINA DE DECREMENTO DEL DATO EN 5*
SALIR   LDAA #$04
        STAA TFLG1      *LIMPIAR BANDERA DE IC1*
        LDAA #$88       *HABILITAR INTERRUPCIONES*
        TAP
        RTI

*********************************
*SUBRUTINA DE ATENCION A TECLA1*
*********************************
PROG1   PSHA            *SUBRUTINA DE INCREMENTO DEL DATO EN 1*
        INC DATO
        JSR PRESENT     *LLAMADO A SUBRUTINA DE CONVERSION Y ENVIO DEL*
        PULA            *DATO A DISPLAY*
        RTS

*********************************
*SUBRUTINA DE ATENCION A TECLA2*
*********************************
PROG2   PSHA            *SUBRUTINA DE DECREMENTO DEL DATO EN 1*
        DEC DATO
        JSR PRESENT     *LLAMADO A SUBRUTINA DE CONVERSION Y ENVIO DEL*
        PULA            *DATO A DISPLAY*
        RTS

*********************************
*SUBRUTINA DE ATENCION A TECLA3*
*********************************
PROG3   PSHA            *SUBRUTINA DE INCREMENTO DEL DATO EN 5*
        INC DATO
        INC DATO
        INC DATO
        INC DATO
        INC DATO
        JSR PRESENT     *LLAMADO A SUBRUTINA DE CONVERSION Y ENVIO DEL*
        PULA            *DATO A DISPLAY*
        RTS


*********************************
*SUBRUTINA DE ATENCION A TECLA4*
*********************************
PROG4   PSHA            *SUBRUTINA DE DECREMENTO DEL DATO EN 5*
        DEC DATO
        DEC DATO
        DEC DATO
        DEC DATO
        DEC DATO
        JSR PRESENT     *LLAMADO A SUBRUTINA DE CONVERSION Y ENVIO DEL*
        PULA            *DATO A DISPLAY*
        RTS

*********************************************
*SUBRUTINA DE CONVERSION DE DATO HEXADECIMAL A*
*DECIMAL Y ENVIA A DISPLAY ALFANUMERICO      *
*********************************************
```

```
PRESENT PSHX
        PSHB
        PSHA
        LDD #$0000      *ENCERAMOS EL DOBLE ACUMULADOR D (A+B)*
        LDAB DATO       *CARGAMOS EL DOBLE ACUMULADOR D CON EL DATO*
        LDX #100        *CARGAMOS EL REGISTRO X CON #100*
        IDIV            *DIVIDIMOS EL DATO EN CENTENAS*
        XGDX            *INTERCAMBIAMOS REGISTROS D Y X*
        STAB DIGIT      *REGISTRAMOS EL COCIENTE DESDE B A DIGIT*
        LDAA #$05       *DIRECCIONO POSICION 5*
        STAA CURSOR     *REGISTARMOS EL CURSOR EN POSICION 5*
        JSR MCURSOR     *LLAMADO A SUBRUTINA PARA POSICIONAR EL CURSOR*
        JSR PRINTD      *VISUALIZO CENTENAS*
        XGDX            *INTERCAMBIAMOS COCIENTE Y RESIDUO*

        LDX #10
        IDIV
        STAB DIGIT
        LDAA #$07       *DIRECCIONO POSICION 7*
        STAA CURSOR
        JSR MCURSOR
        JSR PRINTD      *VISUALIZ0 UNIDADES*


        XGDX
        STAB DIGIT
        LDAA #$06       *DIRECCIONO POSICION 6*
        STAA CURSOR
        JSR MCURSOR
        JSR PRINTD      *VISUALIZO DECENAS*
        PULA
        PULB
        PULX
        RTS


*************************************************
*SUBRUTINA INICIALIZACION DEL DISPLAY           *
*************************************************
*LA ASIGNACION DE PINES DEL PORTICO B ES:       *
*PB.7  PB.6  PB.5  PB.4  PB.3  PB.2  PB.1 PB.0  *
*NC    E     RS    R/W   D7    D6    D5   D4    *
*                        D3    D2    D1   D0    *
*************************************************
*                       INICIALIZACION
*
INICDIS LDY #PORTB      *CARGAMOS EL REGIST. Y CON LA DIRECCION DEL PORTICO
B*
        BCLR 0,Y #$40   *ENCERAMOS LA HABILITACION E DE EL DISPLAY*
        LDAA #$43       *CARGAMOS LA INSTRUCCION DE INICIALIZACION*
        STAA PORTB      *REGISTRAMOS LA INICIALIZACION EN EL PORTICO B*
        JSR RETARDO     *LLAMADO A SUBRUTINA PARA FIJAR EL DATO*
        BCLR 0,Y #$40   *ENCERAMOS LA HABILITACION E DEL DISPLAY*

        LDAA #$43
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40
        LDAA #$43
        STAA PORTB
        JSR RETARDO
```

```
        BCLR 0,Y #$40

        LDY #PORTB
        BCLR 0,Y #$40
        LDAA #$42       *TRANSFERENCIA DE 4 BITS PARA EL DISPLAY*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40
        LDAA #$42       *TRANSFERENCIA DE 4 BITS PARA EL DISPLAY*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        LDAA #$48       *HABILITACION DE DOS LINEAS PARA EL DISPLAY*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        LDAA #$40       *ENCENDIDO DEL CURSOR*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        LDAA #$4E       *ENCENDIDO DEL BLINK*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        LDAA #$40       *INCREMENTAR EL CURSOR*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        LDAA #$46       *INCREMENTAR SHIFT OFF*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40
        JSR CLEARD
        RTS

*****************************************
*ENVIO DE MENSAJE AL DISPLAY V&W-HC11 OK *
*****************************************

SEND    JSR CLEARD
        LDX #MENS1      *CARGAMOS EN X LA DIRECCION DE INICIO DE MENS1*
        LDAB #$00       *              V&W-HC11                       *

FILA1   LDAA 0,X        *CARGAMOS EN A EL BYTE MAS SIGNIFICATIVO*
        BCLR 0,Y #$40
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        INX
        LDAA 0,X        *CARGAMOS EN A EL BYTE MENOS SIGNIFICATIVO*
        BCLR 0,Y #$40
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40
```

```
        INX
        INCB
        CMPB #$10        *COMPARAMOS CON EL #10 PARA LLENAR PRIMERA FILA*
        BNE FILA1

        LDAA #$4C       *DIRECCIONO SEGUNDA FILA BYTE MAS SIGNIFICATIVO*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40
        LDAA #$40       *DIRECCIONO SEGUNDA FILA BYTE MENOS SIGNIFICATIVO*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40



        LDX #MENS2      *CARGAMOS EN X LA DIRECCION DE INICIO DE MENS2*
        LDAB #$40       *               O.K                           *
FILA2   LDAA 0,X        *CARGAMOS EN A EL BYTE MAS SIGNIFICATIVO      *
        BCLR 0,Y #$40
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        INX
        LDAA 0,X        *CARGAMOS EN A EL BYTE MENOS SIGNIFICATIVO*
        BCLR 0,Y #$40
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        INX
        INCB
        CMPB #$50       *COMPARAMOS CON #50 PARA EL LIMITE SUPERIOR DE EL *
        BNE FILA2       *             DISPLAY                             *
        JSR RETAR2
        JSR RETAR2
        JSR RETAR2      *RETARDOS PARA PRESENTACION DE MENSAJE            *
        JSR RETAR2
        JSR RETAR2
        JSR RETAR2
        RTS

****************************************
*SUBRUTINA DE RETARDO PARA FIJAR EL DATO*
****************************************

RETARDO PSHX
        LDX #$FF        *RETARDO DE 127 MICROSEGUNDOS*
RETARD1 DEX
        BNE RETARD1
        PULX
        RTS

************************************************
*SUBRUTINA PARA APAGADO Y PARPADEO DEL CURSOR  *
************************************************

BLINK0
        BCLR 0,Y #$40
```

```
        LDAA #$40       *COMANDO PARA APAGADO DEL CURSOR *
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        LDAA #$4C       *COMANDO PARA APAGADO DEL PARPADEO *
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40
        RTS

*********************************
*SUBRUTINA PARA BORRAR EL DISPLAY*
*********************************

CLEARD  BCLR 0,Y #$40
        LDAA #$40       *COMANDO PARA BORRAR EL BYTE MAS SIGNIFICATIVO*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        LDAA #$41       *COMANDO PARA BORRAR EL BYTE MENOS SIGNIFICATIVO*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40
        JSR RETARDO
        RTS

*********************************************
*SUBRUTINA DE RETARDO PARA PRESENTAR EL DATO*
*********************************************

RETAR2  PSHY
        LDY #$FFFF      *RETARDO DE .32 MILISEGUNDOS*
RETARD2 DEY
        BNE RETARD2
        PULY
        RTS

*************************************************************
*  SUBRUTINA PARA  POSICIONAR EL CURSOR EN EL DISPLAY       *
*INGRESAR LA DIRECCION EN FORMATO HEXADECIMAL EN LA LOCALIDAD*
*        CURSOR DESDE 00 HASTA 0F (FILA 1)                  *
*************************************************************

MCURSOR PSHX
        PSHB
        PSHA
        LDAA CURSOR     *CARGAMOS EN A LA DIRECCION DEL CURSOR*
        ANDA #$10
        CMPA #$10       *COMPARAMOS CON #10 PARA UBICAR EN LA PRIMERA FILA*
        BNE ENVIO
        RTS
ENVIO   LDAA CURSOR     *CARGAMOS EN A Y B LA DIRECCION DEL CURSOR*
        LDAB CURSOR
        ORAA #$80       *INCLUIMOS LA HABILITACION E*
        ORAB #$80
        LSRA
        LSRA
        LSRA
        LSRA
```

```
        ORAA #$40
        BCLR 0,Y #$40
        STAA PORTB      *ENVIO LA DIRECCION ALTA*
        JSR RETARDO
        BCLR 0,Y #$40

        ANDB #$0F       *ENVIO LA DIRECCION BAJA*
        ORAB #$40
        BCLR 0,Y #$40
        STAB PORTB
        JSR RETARDO
        BCLR 0,Y #$40
        PULA
        PULB
        PULX
        RTS

*****************************************************************
*SUBRUTINA PARA ESCRIBIR UNO O VARIOS CARACTERES EN LA POSICION*
*                    ESPECIFICADA POR EL CURSOR                 *
*****************************************************************

SENDM   PSHX
        PSHA
        PSHB
SIGUE   LDAA 0,X
        LDAB 0,X
        CMPA #EOT       *COMPARAMOS EL CODIGO ASCII CON EL #4*
        BEQ FINTEX

        LSRA            *ENVIO LOS PRIMEROS 4 BITS*
        LSRA
        LSRA
        LSRA
        ORAA #$60
        BCLR 0,Y #$40
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40


        ANDB #$0F        *ENVIA LOS SEGUNDOS 4 BITS*
        ORAB #$60
        BCLR 0,Y #$40
        STAB PORTB
        JSR RETARDO
        BCLR 0,Y #$40
        INX
        JMP SIGUE
FINTEX  PULB
        PULA
        PULX
        RTS

*****************************************************************
*SUBRUTINA PARA ESCRIBIR UN NUMERO  DE 0 A 9 EN LA DIRECCION    *
*                    ESPECIFICADA POR EL CURSOR                 *
*****************************************************************

PRINTD  PSHX
        PSHB
```

```
        PSHA
        LDAA DIGIT      *CARGAMOS A Y B CON EL DIGITO*
        LDAB DIGIT
        ADDA #$30
        ADDB #$30

        LSRA            *ENVIO LOS PRIMEROS 4 BITS*
        LSRA
        LSRA
        LSRA
        ORAA #$60
        BCLR 0,Y #$40
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        ANDB #$0F        *ENVIO LOS SEGUNDOS 4 BITS*
        ORAB #$60
        BCLR 0,Y #$40
        STAB PORTB
        JSR RETARDO
        BCLR 0,Y #$40
        PULA
        PULB
        PULX
        RTS

**********
*MENSAJES*
**********

MENS1   FCC 'b`b`b`b`efbfegk`dhdccacab`b`b`b`'
        FCB EOT

MENS2   FCC 'b`b`b`b`b`b`b`c`dkb`b`b`b`b`b`b`'
        FCB EOT

TEXTO   FCC 'DATO='
        FCB EOT



*****
*END*
*****
```

```
*********************************
*PROGRAMA PRACTICA 6 NUMERAL 6.1*
*********************************
TEX     EQU $15    *LOCALIDADES DE RAM PARA GUARDAR EL TEXTO*
TEX1    EQU $16
CURSOR  EQU $20    *LOCALIDAD DE RAM PARA GUARDAR LA POSICION DEL CURSOR*
EOT     EQU $04
RAMI    EQU 30
PORTB   EQU #$1004 *DIRECCION DEL PORTICO B*
SCCR1   EQU #$102C *DIRECCION DEL REGISTRO QUE DEFINE EL FORMATO DE
TRANSMISION SCI*
SCCR2   EQU #$102D *DIRECCION DEL REGISTRO QUE HABILITA INTERRUPCIONES POR
SCI*
SCSR    EQU #$102E *DIRECCION DEL REISTRO QUE INDICA EL ESTADO DE SCI*
BAUD    EQU #$102B *DIRECCION DEL REGISTRO QUE DEFINE LA VELOCIDAD DEL SCI*
SCDR    EQU #$102F *DIRECCION DEL REGISTRO QUE ALMACENA EL DATO A SER
TRANSMITIDO/RECIBIDO*
OPTION  EQU #$1039 *DIRECCION DEL REGISTRO DE HABILITACION DEL CONVERSOR
ADC*
ADCTL   EQU #$1030 *DIRECCION DEL REGISTRO DE CONTROL DEL CONNVERSOR ADC*
ADR1    EQU #$1031 *DIRECCION DEL REGISTRO DE DATOS DEL CONVERSOR ADC*
TECLADO EQU #$6000
TMSK1   EQU #$1022 *DIRECCION DEL REGISTRO DE MASCARAS PARA IC2*
TFLG1   EQU #$1023 *DIRECCION DEL REGISTRO DE BANDERAS DE IC2*
TCTL2   EQU #$1021 *DIRECCION DEL REGISTRO QUE GENERA LA INTERRUPCION INTC2
(FLANCO)*
LOCHEX  EQU #$D000 *LOCALIDADES DE RAM PARA GUARDAR EL VALOR DE LA
VELOCIDAD*
LOCHEX1 EQU #$D001
LOCDEC  EQU #$D002
LOCDEC1 EQU #$D003
LOCDEC2 EQU #$D004
LOCDEC3 EQU #$D005
MULTI   EQU #$D006 *LOCALIDADES DE RAM PARA GUARDAR VALORES DE OPERACIONES
INTERMEDIAS*
DIVI    EQU #$D007
DIVI1   EQU #$D008

        ORG $00E5
        JMP INTIC2  *SALTO A SUBRUTINA DE INTERRUPCION INPUT COMPARE 2*

        ORG $C000
*****************************
*INICIALIZACION DEL PROGRAMA*
*****************************
        LDAA #$20
        STAA TECLADO
        LDAA #$08
        STAA SCCR1 *COMUNICACION 1-8-1*
        LDAA #$08
        STAA SCCR2 *HABILITA TRANSMISION *
        LDAA #$30
        STAA BAUD  *SETEA VELOCIDAD A 9600 BPS*
        LDAA #$93
        STAA OPTION *HABILITA EL CONVERSOR ADC*
        LDAA #$08
        STAA TCTL2  *PARA QUE INTERRUPCION INTIC2 SE GENERE POR FLANCO
NEGATIVO*
```

```
        LDAA #$02
        STAA TMSK1  *QUITAR MASCARA A IC2 *
        LDAA #$01
        STAA $4000 *HABILITAR FLIP-FLOP*
        LDAA #$04
HABIL   STAA SCCR2 *HABILITAR RECEPCION*
        LDX #SCSR
ESPE    BRCLR 0,X #$20 ESPE *ESPERA POR RECEPCION DE DATO COMPLETO*
        LDAA SCSR
        LDAA SCDR
        CMPA #$01
        BNE HABIL

        LDAA #$88
        TAP          *HABILITAR INTERRUPCIONES*

********************
*PROGRAMA PRINCIPAL*
********************
        CLR MULTI
        CLR DIVI
        CLR DIVI1
        CLR TEX
        CLR TEX1
        LDAA #251
        STAA MULTI     *CARGAR LOS REGISTROS MULTI Y DIVI PARA EL
ESCALAMIENTO*
        LDAA #16
        STAA DIVI1
        LDAA #$21      *FIJAR MODO DE TRABAJO DEL ADC: ENTRADA AN1 CONVERSION
NO CONTINUA*
        STAA ADCTL

        JSR INICDIS
        JSR BLINK0
        JSR CLEARD
        LDX #TEXTO
        JSR SENDM

        LDX #TEXTO1
        STX TEX


PROGPRI LDAA ADR1      *CARGO EL VALOR DEL CONVERSOR ADC*
        LDAB MULTI     *OPERACIONES PARA ESCALAR EL VALOR EN RPM O RAD/S*
        MUL
        LDX DIVI
        IDIV
        XGDX
        STAB LOCHEX1   *GUARDO RESULTADOS HEXADECIMALES DE LAS OPERACIONES*
        STAA LOCHEX
        JSR CONVER
        LDAA LOCDEC1   *CARGAR DATO  DE  U. MIL A SER TRANSMITIDO AL
COMPUTADOR*
        ANDA #$0F
        JSR TRANSMI
        LDAA LOCDEC2   *CARGAR DATO  DE  CENTENAS A SER TRANSMITIDO AL
COMPUTADOR*
        ANDA #$0F
        JSR TRANSMI
```

```
        LDAA LOCDEC3  *CARGAR DATO  DE  DECENAS A SER TRANSMITIDO AL
COMPUTADOR*
        ANDA #$0F
        JSR TRANSMI
        JSR LTN211
        JMP PROGPRI

*********************************
*SUBRUTINA DE ATENCION A TECLAD0*
*********************************
INTIC2  PSHX
        PSHB
        PSHA
        PSHY
        LDAA TECLADO
        STAA $6000
        CMPA #$00
        BNE TECLA2
        JSR PROG1
        JMP SALIR
TECLA2  CMPA #$01
        BNE SALIR
        JSR PROG2
SALIR   LDAA #$02
        STAA TFLG1    *LIMPIAR BANDERA DE IC2*
        LDAA #$88
        TAP           *HABILITO INTERRUPCION*
        PULY
        PULA
        PULB
        PULX
        RTI

*********************************
*SUBRUTINA DE ATENCION A TECLA1*
*********************************
PROG1   PSHA

        LDX #TEXTO1
        STX TEX

        LDAA #251
        STAA MULTI     *CARGAR LOS REGISTROS MULTI Y DIVI PARA EL
ESCALAMIENTO*
        LDAA #16
        STAA DIVI1
        PULA
        RTS

*********************************
*SUBRUTINA DE ATENCION A TECLA2*
*********************************
PROG2   PSHA

        LDX #TEXTO2
        STX TEX

        LDAA #248
        STAA MULTI     *CARGAR LOS REGISTROS MULTI Y DIVI PARA EL
ESCALAMIENTO*
        LDAA #151
```

```
        STAA DIVI1
        PULA
        RTS


*******************************************************
*SUBRUTINA DE CONVERSION DE DATO HEXADECIMAL A DECIMAL*
*******************************************************

CONVER  PSHX
        PSHB
        PSHA
        LDAA LOCHEX
        LDAB LOCHEX1
        LDX #10000
        IDIV
        XGDX
        STAB LOCDEC  *LOCALIDAD DE LAS DECENAS DE MIL*
        XGDX
        LDX #1000
        IDIV
        XGDX
        STAB LOCDEC1 *LOCALIDAD DE LAS UNIDADES  DE MIL*
        XGDX
        LDX #100
        IDIV
        XGDX
        STAB LOCDEC2 *LOCALIDAD DE LAS CENTENAS*
        XGDX
        LDX #10
        IDIV
        XGDX
        STAB LOCDEC3  *LOCALIDAD DE LAS DECENAS*
        PULX
        PULB
        PULA
        RTS

*************************************************
*SUBRUTINA PARA TRANSMITIR EL DATO AL COMPUTADOR*
*************************************************
TRANSMI PSHX
        PSHA
        PSHB
        LDX #SCSR
        LDAB #$08
        STAB SCCR2 *HABILITA TRANSMISION*
        STAA SCDR
ES      BRCLR 0,X #$40 ES  *ESPERA MIENTRAS NO SE HAYA TERMINADO LA TRANS.
DEL DATO ANTERIOR*
        LDAB SCSR   *INSTRUCCIONES PARA LIMPIAR LA BANDERA DE INTERRUPCION *
        STAB SCDR   *POR FIN DE TRANSMISION*
        LDAB #00
        STAB SCCR2  *DESHABILITA TRANSMISION*
        PULB
        PULA
        PULX
        RTS

*************************************************
```

```
*SUBRUTINA PARA MANDAR DATOS A DISPLAY ALFANUMERICO*
*****************************************************
LTN211  PSHX
        PSHA
        PSHB
        LDAA #$04
        STAA CURSOR
        JSR MCURSOR
        LDAA LOCDEC1     *ENVIO A DISPLAY UNIDADES DE MIL*
        LDAB LOCDEC1
        JSR PRINTD

        LDAA LOCDEC2     *ENVIO A DISPLAY UNIDADES DE CENTENAS*
        LDAB LOCDEC2
        JSR PRINTD

        LDAA LOCDEC3     *ENVIO A DISPLAY UNIDADES DE DECENAS*
        LDAB LOCDEC3
        JSR PRINTD

        LDAA #$00
        LDAB #$00
        JSR PRINTD

        LDAA #$08
        STAA CURSOR
        JSR MCURSOR
        LDX TEX
        JSR SENDM

        PULB
        PULA
        PULX
        RTS
*************************************************
*SUBRUTINA INICIALIZACION DEL DISPLAY           *
*************************************************
*LA ASIGNACION DE PINES DEL PORTICO B ES:       *
*PB.7  PB.6  PB.5  PB.4  PB.3  PB.2  PB.1 PB.0  *
*NC    E     RS    R/W   D7    D6    D5   D4    *
*                        D3    D2    D1   D0    *
*************************************************
*                        INICIALIZACION
*
INICDIS LDY #PORTB
        BCLR 0,Y #$40   *ENCERAMOS LA HABILITACION E DEL DISPLAY*
        LDAA #$43       *CARGAMOS LA INSTRUCCION DE INICIALIZACION*
        STAA PORTB      *REGISTRAMOS LA INICIALIZACION EN EL PORTICO B*
        JSR RETARDO     *LLAMADO A SUBRUTINA PARA FIJAR EL DATO*
        BCLR 0,Y #$40   *ENCERAMOS LA HABILITACION E DEL DISPLAY*

        LDAA #$43
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40
        LDAA #$43
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        LDY #PORTB
```

```
        BCLR 0,Y #$40
        LDAA #$42       *TRANSFERENCIA DE 4 BITS PARA EL DISPLAY*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40
        LDAA #$42       *TRANSFERENCIA DE 4 BITS PARA EL DISPLAY*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        LDAA #$48       *HABILITACION DE DOS LINEAS PARA EL DISPLAY*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        LDAA #$40       *ENCENDIDO DEL CURSOR*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        LDAA #$4E       *ENCENDIDO DEL BLINK*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        LDAA #$40       *INCREMENTAR EL CURSOR*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        LDAA #$46       *INCREMENTAR SHIFT OFF*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40
        JSR CLEARD
        RTS

****************************************
*SUBRUTINA DE RETARDO PARA FIJAR EL DATO*
****************************************

RETARDO PSHX
        LDX #$FF        *RETARDO DE 127 MICROSEGUNDOS*
RETARD1 DEX
        BNE RETARD1
        PULX
        RTS

***********************************************
*SUBRUTINA PARA APAGADO Y PARPADEO DEL CURSOR *
***********************************************

BLINK0
        BCLR 0,Y #$40
        LDAA #$40       *COMANDO PARA APAGAR EL CURSOR *
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        LDAA #$4C       *COMANDO PARA DESACTIVAR EL PARPADEO*
        STAA PORTB
```

```
        JSR RETARDO
        BCLR 0,Y #$40
        RTS

*********************************
*SUBRUTINA PARA LIMPIAR EL DISPLAY*
*********************************

CLEARD  BCLR 0,Y #$40
        LDAA #$40       *COMANDO PARA LIMPIAR EL BYTE MAS SIGNIFICATIVO*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        LDAA #$41       *COMANDO PARA EL LIMPIAR EL BYTE MENOS
SIGNIFICATIVO*
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40
        JSR RETARDO
        RTS

*******************************************
*SUBRUTINA DE RETARDO PARA PRESENTAR EL DATO*
*******************************************

RETAR2  PSHY
        LDY #$FFFF      *RETARDO DE .32 MILISEGUNDOS*
RETARD2 DEY
        BNE RETARD2
        PULY
        RTS

*************************************************************
*  SUBRUTINA PARA POSICIONAR EL CURSOR EN EL DISPLAY        *
*INGRESAR LA DIRECCION EN FORMATO HEXADECIMAL EN LA LOCALIDAD*
*        CURSOR DESDE 00 HASTA 0F (FILA 1)                  *
*************************************************************
MCURSOR PSHX
        PSHB
        PSHA
        LDAA CURSOR     *CARGAMOS EN A LA DIRECCION DEL CURSOR*
        ANDA #$10
        CMPA #$10       *COMPARAMOS CON #10 PARA UBICAR EN LA PRIMERA FILA*
        BNE ENVIO
        RTS
ENVIO   LDAA CURSOR     *CARGAMOS EN A Y B LA DIRECCION DEL CURSOR*
        LDAB CURSOR
        ORAA #$80       *INCLUIMOS LA HABILITACION E*
        ORAB #$80
        LSRA
        LSRA
        LSRA
        LSRA
        ORAA #$40
        BCLR 0,Y #$40
        STAA PORTB      *ENVIO LA DIRECCION ALTA*
        JSR RETARDO
        BCLR 0,Y #$40

        ANDB #$0F       *ENVIO LA DIRECCION BAJA*
```

```
        ORAB #$40
        BCLR 0,Y #$40
        STAB PORTB
        JSR RETARDO
        BCLR 0,Y #$40
        PULA
        PULB
        PULX
        RTS

*****************************************************************
*SUBRUTINA PARA ESCRIBIR UNO O VARIOS CARACTERES EN LA POSICION*
*                    ESPECIFICADA POR EL CURSOR                 *
*****************************************************************
SENDM   PSHX
        PSHA
        PSHB            *CARGAMOS EN X LA POSICION DEL TEXTO*
SIGUE   LDAA 0,X
        LDAB 0,X
        CMPA #EOT       *COMPARAMOS EL CODIGO ASCII CON EL #4*
        BEQ FINTEX

        LSRA            *ENVIO LOS PRIMEROS 4 BITS*
        LSRA
        LSRA
        LSRA
        ORAA #$60
        BCLR 0,Y #$40
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40


        ANDB #$0F        *ENVIA LOS SEGUNDOS 4 BITS*
        ORAB #$60
        BCLR 0,Y #$40
        STAB PORTB
        JSR RETARDO
        BCLR 0,Y #$40
        INX
        JMP SIGUE
FINTEX  PULB
        PULA
        PULX
        RTS

*****************************************************************
*SUBRUTINA PARA ESCRIBIR UN NUMERO  DE 0 A 9 EN LA DIRECCION    *
*                    ESPECIFICADA POR EL CURSOR                 *
*****************************************************************

PRINTD  PSHX
        PSHB
        PSHA
        ADDA #$30
        ADDB #$30

        LSRA            *ENVIO LOS PRIMEROS 4 BITS*
        LSRA
        LSRA
        LSRA
```

```
        ORAA #$60
        BCLR 0,Y #$40
        STAA PORTB
        JSR RETARDO
        BCLR 0,Y #$40

        ANDB #$0F        *ENVIO LOS SEGUNDOS 4 BITS*
        ORAB #$60
        BCLR 0,Y #$40
        STAB PORTB
        JSR RETARDO
        BCLR 0,Y #$40
        PULA
        PULB
        PULX
        RTS

**********
*MENSAJES*
**********

TEXTO   FCC 'VEL='
        FCB EOT
TEXTO1  FCC '(RPM)    '
        FCB EOT
TEXTO2  FCC '(RAD/SG)'
        FCB EOT

*****
*END*
*****
```

```
*********************************
*PROGRAMA PRACTICA 7 NUMERAL 4.a*
*********************************

SCCR1   EQU #$102C *DIRECCION DEL REGISTRO QUE DEFINE EL FORMATO DE
TRANSMISION SCI*
SCCR2   EQU #$102D *DIRECCION DEL REGISTRO QUE HABILITA INTERRUPCIONES POR
SCI*
SCSR    EQU #$102E *DIRECCION DEL REGISTRO QUE INDICA EL ESTADO DE SCI*
BAUD    EQU #$102B *DIRECCION DEL REGISTRO QUE DEFINE LA VELOCIDAD DEL SCI*
SCDR    EQU #$102F *DIRECCION DEL REGISTRO QUE ALMACENA EL DATO RECIBIDO*
DAC     EQU #$6400

        ORG $00C4
        JMP INTSCI

        ORG $C000

************************************************
*INICIALIZACION DE REGISTROS USADOS EN EL PROGRAMA*
************************************************

        LDAA #$08
        STAA SCCR1 *COMUNICACION 1-8-1*
        LDAA #$30
        STAA BAUD  *SETEA VELOCIDAD A 9600 BPS*
        LDAB #$24
        STAB SCCR2 *HABILITA RECEPCION E INTERRUPCION DE RECEPCION*
        LDAA #$01  *HABILITO FLIP-FLOP*
        STAA $4000
        CLR DAC    *ENCERAR SALIDA DEL CONVERSOR*
        LDAA #$88
        TAP        *HABILITAR INTERRUPCIONES*

********************
*PROGRAMA PRINCIPAL*
********************

PROGPRI JMP PROGPRI

*****************************************
*SUBRUTINA DE INTERRUPCION DE RECEPCION    *
*****************************************
INTSCI  LDAA SCDR   *LECTURA DE DATO*
        STAA DAC    *ENVIAR DATO AL CONVERSOR D/A*
        LDAB SCSR   *INSTRUCIONES PARA LIMPIAR LA BANDERA DE INTERRUPCION
POR FIN DE TRANSMISION*
        LDAB SCDR
        RTI

*****
*END*
*****
```

```
*********************************
*PROGRAMA PRACTICA 7 NUMERAL 4.b*
*********************************

HPRIO   EQU #$103C *DIRECCION DEL REGISTRO QUE DEFINE PRIORIDAD DE
INTERRUPCIONES*
SCCR1   EQU #$102C *DIRECCION DEL REGISTRO QUE DEFINE EL FORMATO DE
TRANSMISION SCI*
SCCR2   EQU #$102D *DIRECCION DEL REGISTRO QUE HABILITA INTERRUPCIONES POR
SCI*
SCSR    EQU #$102E *DIRECCION DEL REGISTRO QUE INDICA EL ESTADO DE SCI*
BAUD    EQU #$102B *DIRECCION DEL REGISTRO QUE DEFINE LA VELOCIDAD DEL SCI*
SCDR    EQU #$102F *DIRECCION DEL REGISTRO QUE ALMACENA EL DATO RECIBIDO*
TCNT    EQU #$100E *DIRECCION DEL REGISTRO DEL TIMER/COUNTER*
TOC2    EQU #$1018 *DIRECCION DEL REGISTRO DEL OUTPUT COMPARE 2*
TMSK1   EQU #$1022 *DIRECCION DEL REGISTRO DE MASCARAS PARA OC2*
TFLG1   EQU #$1023 *DIRECCION DEL REGISTRO DE BANDERAS DE OC2*
PERIODO EQU #$D000
ALTO    EQU #$D002
ALTO1   EQU #$D003
BAJO    EQU #$D004
BAJO1   EQU #$D005
SALDIG  EQU #$6800
AUX     EQU #$0000 *DIRECCION AUXILIAR  PARA SALDIG*

        ORG $00DC
        JMP INTOC2
        ORG $00C4
        JMP INTSCI

        ORG $C000

*************************************************
*INICIALIZACION DE REGISTROS USADOS EN EL PROGRAMA*
*************************************************

        LDAA #$08
        STAA SCCR1 *COMUNICACION 1-8-1*
        LDAA #$30
        STAA BAUD  *SETEA VELOCIDAD A 9600 BPS*
        LDAB #$24
        STAB SCCR2 *HABILITA RECEPCION E INTERRUPCION DE RECEPCION*
        LDAA #$01       *HABILITO FLIP-FLOP*
        STAA $4000
        LDAA #$09
        STAA HPRIO   *DEFINE INTOC2 COMO LA INTERRUPCION DE MAS ALTA
PRIORIDAD*
        LDD #$0001
        STD ALTO     *INICIALIZA EL  REGISTRO DEL ANCHO DEL PULSO EN CERO*
        LDD #$0000
        STD BAJO     *INICIALIZA EL  REGISTRO DEL NIVEL BAJO  EN CERO*
        LDAA #00
        STAA SALDIG  *INICIALIZA LA SALIDA DIGITAL EN CERO*
        STAA AUX     *INICIALIZO REGISTRO AUXILIAR*
        LDD TCNT
        ADDD #$00FF
        STD TOC2     *INICIALIZA EL VALOR DE RETARDO PARA LA INTERRUPCION
TOC2*
        LDAA #$40
        STAA TMSK1   *HABILITA INTERRUPCION DE TIMER OUPUT COMPARE 2*
```

```
        LDAA #$88
        TAP         *HABILITAR INTERRUPCIONES*

*******************
*PROGRAMA PRINCIPAL*
*******************
PROGPRI JMP PROGPRI

*****************************************
*SUBRUTINA DE INTERRUPCION DE RECEPCION    *
*****************************************
INTSCI  LDAA SCDR   *LECTURA DE DATO*
        LDAB SCSR   *INSTRUCIONES PARA LIMPIAR LA BANDERA DE INTERRUPCION
POR FIN DE TRANSMISION*
        LDAB SCDR
        STAA ALTO1
        RTI

*****************************************
*SUBRUTINA DE INTERRUPCION DE TEMPORIZACION*
*****************************************
INTOC2  LDAA #$40
        STAA TFLG1
        LDY #SALDIG
        LDX #AUX
        BRCLR 0,X #$01 UNO      *COMPARACION DEL ESTADO ANTERIOR DE LA
SALIDA DIGITAL*
        CLR SALDIG           *ENCERO LA SALIDA DIGITAL*
        CLR AUX           *ENCERO EL REGISTRO AUXILIAR DE LA SALIDA DIGITAL*
        LDAA #$FF
        SUBA ALTO1
        STAA BAJO1
        LDD TCNT
        ADDD BAJO1
        STD TOC2
        JMP SIGA1
UNO     BSET 0,Y #$01           *UNO LOGICO A LA SALIDA DIGITAL*
        BSET 0,X #$01           *UNO LOGICO A EL REGISTRO AUXILIAR DE LA
SALIDA DIGITAL*
        LDD TCNT
        ADDD ALTO1
        STD TOC2
SIGA1   RTI
```

```
*******************************
*PROGRAMA PRACTICA 8 NUMERAL 4*
*******************************
HPRIO   EQU #$103C *DIRECCION DEL REGISTRO QUE DEFINE PRIORIDAD DE
INTERRUPCIONES*
SCCR1   EQU #$102C *DIRECCION DEL REGISTRO QUE DEFINE EL FORMATO DE
TRANSMISION SCI*
SCCR2   EQU #$102D *DIRECCION DEL REGISTRO QUE HABILITA INTERRUPCIONES POR
SCI*
SCSR    EQU #$102E *DIRECCION DEL REGISTRO QUE INDICA EL ESTADO DE SCI*
BAUD    EQU #$102B *DIRECCION DEL REGISTRO QUE DEFINE LA VELOCIDAD DEL SCI*
SCDR    EQU #$102F *DIRECCION DEL REGISTRO QUE ALMACENA EL DATO TRANSMITIDO
O RECIBIDO*
TCNT    EQU #$100E *DIRECCION DEL REGISTRO DEL TIMER/COUNTER*
TMSK1   EQU #$1022 *DIRECCION DEL REGISTRO DE MASCARAS PARA IC1*
TFLG1   EQU #$1023 *DIRECCION DEL REGISTRO DE BANDERAS DE IC1*
OPTION  EQU #$1039 *DIRECCION DEL REGISTRO DE HABILITACION DEL CONVERSOR
ADC*
ADCTL   EQU #$1030 *DIRECCION DEL REGISTRO DE CONTROL DEL CONNVERSOR ADC*
ADR1    EQU #$1031 *DIRECCION DEL REGISTRO DE DATOS DEL CONVERSOR ADC*
DAC     EQU $6400
SETPOIN EQU $00
DATRANS EQU $01
VEL     EQU $02
VOLTA   EQU $03
TEC     EQU $10
TECLAS  EQU #$6000

        ORG $00E8
        JMP INTIC1      *SALTO A SUBRUTINA DE ATENCION A TECLAS*

        ORG $00C4
        JMP INTSCI      *SALTO A SUBRUTINA DE COMUNICACION SERIAL*

        ORG $C000

************************************************
*INICIALIZACION DE REGISTROS USADOS EN EL PROGRAMA*
************************************************

        LDAA #$AF
        STAA SETPOIN
        CLR DATRANS
        CLR VEL
        LDAA #$7F
        STAA VOLTA
        LDAA #$04
        STAA HPRIO
        LDAA #$08
        STAA SCCR1 *COMUNICACION 1-8-1*
        LDAA #$30
        STAA BAUD  *SETEA VELOCIDAD A 9600 BPS*
        LDAB #$24
        STAB SCCR2 *HABILITA RECEPCION Y SU INTERRUPCION*
        LDAA #$01       *HABILITO FLIP-FLOP*
        STAA $4000
        LDAA #$04
        STAA TMSK1   *HABILITA INTERRUPCION DE TECLAS INPUT CAPTURE 1*
        LDAA #$93
        STAA OPTION *HABILITA EL CONVERSOR ADC*
```

```
        LDAA #$01                *FIJAR MODO DE TRABAJO DEL ADC: ENTRADA AN1
CONVERSION  NO CONTINUA*
        STAA ADCTL
        LDAA #$88
        TAP          *HABILITAR INTERRUPCIONES*

********************
*PROGRAMA PRINCIPAL*
********************
PROGPRI JSR DISPLAY
        JSR LEEVEL
        JSR ESCVOL
        JMP PROGPRI

*****************************************
*SUBRUTINA DE INTERRUPCION DE RECEPCION    *
*****************************************
INTSCI  LDAA #$98
        TAP          *DESHABILITAR INTERRUPCIONES*
        LDAA SCSR    *INSTRUCIONES PARA LIMPIAR LA BANDERA DE REGISTRO DE
RECEPCION LLENO*
        LDAA SCDR
        CMPA #$01
        BNE DOS
        LDAB VEL
        STAB DATRANS
        JSR TRANSMI
        LDAB SETPOIN
        STAB DATRANS
        JSR TRANSMI
        JMP SALIR
DOS     BNE SALIR
        LDX #SCSR
ESPERA2 BRCLR 0,X #$20 ESPERA2  *ESPERA MIENTRAS NO SE HAYA TERMINADO DE
        LDAA SCSR               *RECIBIE EL SEGUNDO DATO*
        LDAA SCDR               *LIMPIA BANDERA DE REGISTRO DE RECEPCION
LLENO*
        STAA VOLTA
SALIR   LDAA #$88
        TAP          *HABILITAR INTERRUPCIONES*
        RTI

************************************************
*INTERRUPCION DE TECLAS  PARA VARIAR EL SET POINT*
************************************************
INTIC1  LDAA #$98        *DESHABILITO INTERRUPCIONES
        TAP
        LDAA TECLAS      *LECTURA DE TECLA PRESIONADA*
        CMPA #$01
        BNE TECLA2
        LDAA SETPOIN
        ADDA #05
        STAA SETPOIN
        JMP SALIR3
TECLA2  CMPA #$02
        BNE TECLA3
        LDAA SETPOIN
        SUBA #$05
        STAA SETPOIN
        JMP SALIR3
TECLA3  CMPA #$04
```

```
        BNE TECLA4
        LDAA SETPOIN
        ADDA #10
        STAA SETPOIN
        JMP SALIR3
TECLA4  CMPA #$08
        BNE SALIR3
        LDAA SETPOIN
        SUBA #10
        STAA SETPOIN
SALIR3  LDAA #$04
        STAA TFLG1      *LIMPIAR BANDERA DE IC1*
        LDAA #$88
        TAP
        RTI


*********************************************
*SUBRUTINA PARA TRANSMITIR DATOS AL COMPUTADOR*
*********************************************
TRANSMI PSHX
        PSHY
        PSHA
        PSHB
        LDX #SCSR
        LDY #SCCR2
        LDAA DATRANS
        BSET 0,Y #$08           *HABILITAR TRANSMISION*
        STAA SCDR
ESPERA  BRCLR 0,X #$40 ESPERA  *ESPERA MIENTRAS NO SE HAYA TERMINADO LA
TRANSMISION DEL DATO ANTERIOR*
        BCLR 0,Y #$08           *DESHABILITAR TANSMISION*
        LDAB SCSR   *INSTRUCIONES PARA LIMPIAR LA BANDERA DE POR FIN DE
TRANSMISION*
        STAB SCDR
        PULB
        PULA
        PULY
        PULX
        RTS

*********************************************
*SUBRUTINA PARA LEER EL VALOR DE LA VELOCIDAD*
*********************************************
LEEVEL  PSHA
        LDAA ADR1
        STAA VEL
        PULA
        RTS
*****************************************************
*SUBRUTINA PARA SACAR EL VALOR DEL VOLTAJE AL MOTOMATIC*
*****************************************************
ESCVOL  PSHA
        LDAA VOLTA
        STAA DAC
        PULA
        RTS

*****
*END*
*****
```

```
************************
*PROGRAMA PRACTICA No.9*
************************


OPTION  EQU #$1039 *DIRECCION DEL REGISTRO DE HABILITACION DEL CONVERSOR
ADC*
ADCTL   EQU #$1030 *DIRECCION DEL REGISTRO DE CONTROL DEL CONVERSOR ADC*
ADR1    EQU #$1031 *DIRECCION DEL REGISTRO DE DATOS DEL CONVERSOR ADC*
DISPLAY EQU #$6000
SALDIG  EQU #$6800

****************
*INICIALIZACION*
****************
        ORG $C000
        LDAA #$93
        STAA OPTION *HABILITA EL CONVERSOR ADC*
        LDAA #$21   *FIJAR MODO DE TRABAJO DEL ADC: ENTRADA AN1 CONVERSION
NO CONTINUA*
        STAA ADCTL

********************
*PROGRAMA PRINCIPAL*
********************
PROGPRI LDAA ADR1   *CARGAR EL VALOR DEL CONVERSOR ADC*
        STAA SALDIG *SACAR EL VALOR HACIA EL PORTICO PARALELO*
        STAA DISPLAY
        LDX #$20
DECRE   DEX
        BNE DECRE
        JMP PROGPRI


*****
*END*
*****
```

```
************************************************
*PROGRAMA DE QUICK BASIC -PRACTICA 4- NUMERAL 4.2*
************************************************


     DECLARE SUB CARATULA ()
     CLS

     SCREEN 12
     WINDOW (0, 800)-(800, -800)
     VIEW (0, 0)-(639, 450), 1, 20
     VIEW (15, 15)-(624, 435), 9, 20
     VIEW (322, 32)-(608, 318), 3, 3
     VIEW (340, 50)-(590, 300), 14, 4
     CALL CARATULA
I = 0
PSET (0, 0)
CLOSE
OPEN "COM2:9600,N,8,1,RS,CS,DS,CD" FOR RANDOM AS #1
PRINT #1, CHR$(1);
E = ASC(INPUT$(1, 1))

DO:
         DO: LOOP WHILE LOC(1) < 4
         A = ASC(INPUT$(1, 1))
         B = ASC(INPUT$(1, 1))
         C = ASC(INPUT$(1, 1))
         D = ASC(INPUT$(1, 1))
         LOCATE 10, 1
         RPM = A * 1000 + B * 100 + C * 10 + D
         LOCATE 22, 50: PRINT "VELOCIDAD ="; RPM
         LOCATE 20, 43: PRINT "0"
         LOCATE 3, 43: PRINT "4000 [RPM]"
         WINDOW (0, 4000)-(100, 0)
         LINE -(I, RPM), 4
         I = I + 1
         IF I >= 100 THEN
         CLS
         VIEW (340, 50)-(590, 300), 14, 4
         I = 0
         PSET (0, 0)
         END IF
         LOCATE 20, 5: PRINT "PRESIONE LA TECLA ESC PARA SALIR"

LOOP UNTIL INKEY$ = CHR$(27)

END

SUB CARATULA

     LOCATE 3, 8: PRINT "ESCUELA POLITECNICA NACIONAL"
     LOCATE 4, 6: PRINT "FACULTAD DE INGENIERIA ELECTRICA"
     LOCATE 5, 10: PRINT "DEPARTAMENTO DE CONTROL"
     LOCATE 7, 7: PRINT "CONTROL CON MICROPROCESADORES"
     LOCATE 10, 14: PRINT "PRACTICA 4"
```

```
    LOCATE 12, 3: PRINT "INTERFAZ DE COMUNICACION SERIAL RS-232"
    LOCATE 23, 10: PRINT "VINICIO ACOSTA"
    LOCATE 24, 8: PRINT "WILSON PASTUISACA"

END SUB
```

```
**********************************************
*PROGRAMA DE QUICK BASIC -PRACTICA 6-NUMERAL 4*
**********************************************


     DECLARE SUB CARATULA ()
     CLS

     SCREEN 12
     WINDOW (0, 800)-(800, -800)
     VIEW (0, 0)-(639, 450), 1, 20
     VIEW (15, 15)-(624, 435), 9, 20
     VIEW (322, 32)-(608, 318), 3, 3
     VIEW (340, 50)-(590, 300), 14, 4
     CALL CARATULA
CLOSE
I = 0
PSET (0, 0)
OPEN "COM2:9600,N,8,1,RS,CS,DS,CD" FOR RANDOM AS #1
PRINT #1, CHR$(1)
E = ASC(INPUT$(1, 1))
DO:
         DO: LOOP WHILE LOC(1) < 3
         A = ASC(INPUT$(1, 1))
         B = ASC(INPUT$(1, 1))
         C = ASC(INPUT$(1, 1))
         rpm = A * 1000 + B * 100 + C * 10
         LOCATE 22, 50: PRINT "VELOCIDAD ="; rpm
         LOCATE 20, 43: PRINT "0"
         LOCATE 3, 43: PRINT "4000 [RPM]"
         WINDOW (0, 4000)-(100, 0)
         LINE -(I, rpm), 4
         I = I + 1
         IF I >= 100 THEN
         CLS
         VIEW (340, 50)-(590, 300), 14, 4
         I = 0
         PSET (0, 0)
         END IF
         LOCATE 20, 5: PRINT "PRESIONE LA TECLA ESC PARA SALIR"

LOOP UNTIL INKEY$ = CHR$(27)

END

SUB CARATULA

     LOCATE 3, 8: PRINT "ESCUELA POLITECNICA NACIONAL"
     LOCATE 4, 6: PRINT "FACULTAD DE INGENIERIA ELECTRICA"
     LOCATE 5, 10: PRINT "DEPARTAMENTO DE CONTROL"
     LOCATE 7, 7: PRINT "CONTROL CON MICROPROCESADORES"
     LOCATE 10, 14: PRINT "PRACTICA 6"
     LOCATE 12, 5: PRINT "UTILIZACION DE CONVERSORES A/D"
     LOCATE 23, 10: PRINT "VINICIO ACOSTA"
     LOCATE 24, 8: PRINT "WILSON PASTUISACA"

END SUB
```

```
*********************************************************
*PROGRAMA DE QUICK BASIC -PRACTICA 7-NUMERAL 4.a Y 4.b*
*********************************************************



     DECLARE SUB CARATULA ()
     CLS
     SCREEN 12
     WINDOW (0, 800)-(800, -800)

     VIEW (0, 0)-(639, 450), 1, 20
     VIEW (15, 15)-(624, 435), 9, 20
     VIEW (322, 32)-(608, 318), 3, 3


     CALL CARATULA


CLOSE
OPEN "COM2:9600,N,8,1,RS,CD,DS,CS" FOR RANDOM AS #1

DATO = 1
PRINT #1, CHR$(DATO);
DO:
         R$ = INKEY$
         SELECT CASE R$
         CASE "A"
                 DATO = DATO + 1
                 IF DATO <= 254 THEN
                         PRINT #1, CHR$(DATO);
                 END IF
                 IF DATO > 254 THEN
                         DATO = 254
                 END IF
         CASE "D"
                 DATO = DATO - 1
                 IF DATO >= 1 THEN
                         PRINT #1, CHR$(DATO);
                 END IF
                 IF DATO < 1 THEN
                        DATO = 1
                 END IF

         CASE "P"
                 DATO = 1
                 PRINT #1, CHR$(DATO);
         CASE CHR$(27)
                 GOTO 100
         END SELECT
LOOP

100 END

SUB CARATULA

     LOCATE 3, 8: PRINT "ESCUELA POLITECNICA NACIONAL"
     LOCATE 4, 6: PRINT "FACULTAD DE INGENIERIA ELECTRICA"
     LOCATE 5, 10: PRINT "DEPARTAMENTO DE CONTROL"
```

```
    LOCATE 7, 7: PRINT "CONTROL CON MICROPROCESADORES"
    LOCATE 10, 14: PRINT "PRACTICA 7"
    LOCATE 12, 5: PRINT "SINTESIS DE SE¥ALES DIGITALES"
    LOCATE 23, 10: PRINT "VINICIO ACOSTA"
    LOCATE 24, 8: PRINT "WILSON PASTUISACA"
    LOCATE 8, 44: PRINT "ELIJA LA OPCION"
    LOCATE 10, 44: PRINT "A : AUMENTA LA VELOCIDAD"
    LOCATE 12, 44: PRINT "D : DISMINUYE LA VELOCIDAD"
    LOCATE 14, 44: PRINT "P : PARA EL MOTOR"
    LOCATE 22, 44: PRINT "PRESIONE ESC PARA SALIR"
END SUB
```

```
**********************************************
*PROGRAMA DE QUICK BASIC -PRACTICA 8-NUMERAL 4*
**********************************************


DECLARE SUB CARATULA ()
    CLS
    SCREEN 12
    WINDOW (0, 800)-(800, -800)

    VIEW (0, 0)-(639, 450), 1, 20
    VIEW (15, 15)-(624, 435), 9, 20
    VIEW (322, 32)-(608, 318), 3, 3

    E0=0
    E1=0
    E2=0
    U1=0
    U0=0
    KP=1
    KD=0
    KI=0
    T1=1

    CALL CARATULA


CLOSE
OPEN "COM2:9600,N,8,1,RS,CD,DS,CS" FOR RANDOM AS #1

DATO = 1
PRINT #1, CHR$(DATO);
DO:
      PSET (0,0)
      'ADQUISICION DE DATOS
LOCATE 1,30:PRINT "MEDICIONES DE LA VELOCIDAD DE MOTOR"
LOCATE 23,1:PRINT "PRESIONE <S> PARA PARAR"
PASO = 10
T0 = TIMER
FOR X= 0 TO 1000 STEP PASO
T$ = INKEY$
SELECT CASE T$
CASE "P"
      KP = KP + .1
      IF KP > 255 THEN KP = 255
CASE "A"
      KP = KP - .1
      IF KP < 0 THEN KP = 0
CASE "I"
      KI = KI + .1
      IF KI > 255 THEN KI =255
CASE "B"
      KI = KI - .1
      IF KI < 0 THEN KI = 0
CASE "D"
      KD = KD + .1
      IF KD > 255 THEN KD = 255
CASE "C"
```

```
        KD = KD - .1
        IF KD < 0 THEN KD = 0
CASE "S"
        U1 = 0
        U0 = 0
        GOTO FIN
END SELECT

        LOCATE 20, 1: PRINT "SET POINT ="; SETP;
        LOCATE 21, 30: PRINT "X=";X;
        LOCATE 20, 50: PRINT "KP=";KP;
        LOCATE 21, 50: PRINT "KI=";KI;
        LOCATE 22, 50: PRINT "KD=";KD;
        LOCATE 20, 30: PRINT "U0=";U0;

        PRINT #1, CHR$(1);
        T2 = TIMER
        DO: LOOP WHILE LOC(1) = 0 AND (TIMER -T2) < .1
        VELOCIDAD = ASC(INPUT$(1,1))
        T3 = TIMER
        DO: LOOP WHILE LOC(1) = 0 AND (TIMER -T3) < .1
        SETP = ASC(INPUT$(1,1))
'GRAFICAR VELOCIDAD
        LINE -(X,VELOCIDAD)
        'ALGORITMO DE CONTROL
        B0 = KP * (2 * T1 + 2 * KD + KI * T1 ^ 2) / (2 * T1)
        B1 = KP * (KI * T1 ^ 2 - 2 * T1 - 4 * KD) / (2 * T1)
        B2 = KP * KD / T1

        U0 = U1 + B0 * E0 + B1 * E1 + B2 * E2
        U1 = U0
        E2 = U1
        E1 = E0
        E0 = SETP - VELOCIDAD
        IF U0 = < 0 THEN
              U0 = 0
        ELSE
              IF U0 > 255 THEN
                    U0 = 255
              END IF
        END IF
        PRINT #1, CHR$(2); CHR$(U0);
        NEXT X

        T1 = (TIMER -T0) / 1000 *PASO

LOOP

FIN:
        PRINT #1, CHR$(2); CHR$(U0);
        DO:
        LOCATE 23,1: INPUT "DESEA SALIR DEL PROGRAMA (S/N)";R$
        LOOP WHILE R$ <> "S" AND R$ <> "N"
        IF R$ = "N" THEN GOTO INICIO

SUB CARATULA


    LOCATE 4, 6: PRINT "FACULTAD DE INGENIERIA ELECTRICA"
    LOCATE 5, 10: PRINT "DEPARTAMENTO DE CONTROL"
    LOCATE 7, 7: PRINT "CONTROL CON MICROPROCESADORES"
```

```
    LOCATE 10, 14: PRINT "PRACTICA 8"
    LOCATE 12, 5: PRINT "IMPLEMENTACION DIGITAL DE ALGORITMOS DE CONTROL"
    LOCATE 23, 10: PRINT "VINICIO ACOSTA"
    LOCATE 24, 8: PRINT "WILSON PASTUISACA"
    LOCATE 8, 44: PRINT "ELIJA LA OPCION"
    LOCATE 10, 44: PRINT "A : AUMENTA LA VELOCIDAD"
    LOCATE 12, 44: PRINT "D : DISMINUYE LA VELOCIDAD"
    LOCATE 14, 44: PRINT "P : PARA EL MOTOR"
    LOCATE 22, 44: PRINT "PRESIONE ESC PARA SALIR"
END SUB
```

```
*********************************************
*PROGRAMA DE QUICK BASIC -PRACTICA 9-NUMERAL 4*
*********************************************

CONST PI = 3.141593#

    DECLARE SUB CARATULA ()
    CLS
    DIM A(1000)
    SCREEN 12
    WINDOW (0, 800)-(800, -800)

    VIEW (0, 0)-(639, 450), 1, 20
    VIEW (15, 15)-(624, 435), 9, 20
    VIEW (322, 32)-(608, 318), 3, 3
    VIEW (340, 50)-(590, 300), 14, 4
    LINE (0, 0)-(800, 0), 1
    LINE (400, 800)-(400, -800), 1

    CALL CARATULA

 'VALORES INICIALES Y CALCULO DE COEFICIENTES

 Y2 = 0: Y1 = 0: Y = 0: U2 = 0: U1 = 0: U = 0
 T1 = .000025
 K = 2 * PI * 30 / .6423
 A = (1 - EXP(-K * T1) - (K * T1 * EXP(-K * T1)))
 B = (EXP(-2 * K * T1) - EXP(-K * T1) + (K * T1 * EXP(-K * T1)))
 C = -2 * EXP(-K * T1)
 D = EXP(-2 * K * T1)


   DO:
          FOR J = 1 TO 820
                 'LECTURA DE DATOS DEL PORTICO PARALELO
                 IN1 = -((INP(&H37A) AND 1) = 0)
                 IN14 = -((INP(&H37A) AND 2) = 0)
                 IN16 = -((INP(&H37A) AND 4) = 4)
                 IN17 = -((INP(&H37A) AND 8) = 0)
                 IN15 = -((INP(&H379) AND 8) = 8)
                 IN13 = -((INP(&H379) AND 16) = 16)
                 IN12 = -((INP(&H379) AND 32) = 32)
                 IN10 = -((INP(&H379) AND 64) = 64)
                 R = IN1 + IN14 * 2 + IN16 * 4 + IN17 * 8
                 S = IN15 * 16 + IN13 * 32 + IN12 * 64 + IN10 * 128
                 T = R + S


                 'CALCULO DEL ALGORITMO

                 DATO = ABS((T - 128) * 2)
                 U = DATO
                 Y = A * U1 + B * U2 - C * Y1 - D * Y2

                 'ACTUALIZACION DE DATOS

                 Y2 = Y1
                 Y1 = Y
                 U2 = U1
                 U1 = U
```

```
                'GRAFICACION DE SEÑALES

                Yin = (750 / 255) * DATO
                Yout = (750 / 255) * Y
                X = J - 20
                PSET (X, Yin), 4
                PSET (X, Yout), 8
                LOCATE 17, 10: PRINT "X=", J
                LOCATE 18, 10: PRINT "Yin=", DATO
                LOCATE 19, 10: PRINT "Yout=", Y
                LOCATE 20, 5: PRINT "PRESIONE LA TECLA ESC PARA SALIR"
                LOCATE 21, 5: PRINT "A=", A
                LOCATE 22, 5: PRINT "B=", B
                LOCATE 23, 5: PRINT "C=", C
                LOCATE 24, 5: PRINT "D=", D
        NEXT J
        CLS
        VIEW (340, 50)-(590, 300), 14, 4
        LINE (0, 0)-(800, 0), 1
        LINE (400, 800)-(400, -800), 1

     LOOP UNTIL INKEY$ = CHR$(27)

END

SUB CARATULA

    LOCATE 3, 8: PRINT "ESCUELA POLITECNICA NACIONAL"
    LOCATE 4, 6: PRINT "FACULTAD DE INGENIERIA ELECTRICA"
    LOCATE 5, 10: PRINT "DEPARTAMENTO DE CONTROL"
    LOCATE 7, 7: PRINT "CONTROL CON MICROPROCESADORES"
    LOCATE 10, 14: PRINT "PRACTICA 9"
    LOCATE 12, 10: PRINT "FILTROS DIGITALES"
    LOCATE 26, 52: PRINT "VINICIO ACOSTA"
    LOCATE 27, 50: PRINT "WILSON PASTUISACA"

END SUB
```

SET DE INSTRUCCIONES DEL MC68HC11

1.13.1 **ABA** Suma acumulador B a acumulador A

Operación: ACCA ⇐ (ACCA) + (ACCB)

Descripción: Suma los contenidos de los acumuladores A y B, el resultado de esta operación se sitúa en el acumulador A. El contenido del acumulador B no cambia. Las banderas del CCR que son afectadas por ésta instrucción se resumen en el siguiente cuadro:

S	X	H	I	N	Z	V	C
-	-	⇑	-	⇑	⇑	⇑	⇑

1.13.2 **ABX** Suma acumulador B al registro índice X

Operación: IX ⇐ (IX) + (ACCB)

Descripción: Suma el contenido del acumulador B al contenido del registro índice X, considerando el posible carry generado en el byte de bajo orden del registro índice X. El resultado se hubica en el registro índice X, EL acumulador B no es afectado. No hay una instrucción equivalente que permita sumar al registro índice X el contenido del acumulador A. El CCR no se altera.

1.13.3 **ABY** Suma acumulador B al registro índice Y

Operación: IY ⇐ (IY) + (ACCB)

Descripción: Suma el contenido del acumulador B al contenido del registro índice Y, considerando el posible carry generado en el byte de bajo orden del registro índice Y. El resultado se ubica en el registro índice Y, EL acumulador B no es afectado. No hay

una instrucción equivalente que permita sumar al registro índice Y el contenido del acumulador A. El CCR no es afectado.

1.13.4 **ADC** Suma con carry

Operación: ACCX ⇐ (ACCX) + (M) + (C)

Descripción: Suma el bit del carry a la suma de los contenidos de los acumuladores ACCX y el contenido de la memoria M y ubica el resultado en ACCX. Esta instrucción afecta el bit H del CCR.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	⇑	-	⇑	⇑	⇑	⇑

La operación ADC puede ser ADCA(opr) o ADCB(opr).

1.13.5 **ADD** Suma sin carry

Operación: ACCX ⇐ (ACCX) + (M)

Descripción: Suma el contenido del acumulador ACCX y el contenido de la memoria M, y ubica el resultado en ACCX. Esta instrucción afecta el bit H del CCR.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	⇑	-	⇑	⇑	⇑	⇑

La operación ADD puede ser ADDA(opr) o ADDB(opr).

1.13.6 **ADDD** Suma al doble acumulador

Operación: ACCD ⇐ (ACCD) + (M: M+1)

Descripción: Suma el contenido del acumulador ACCD y el contenido de la memoria M concatenado con M+1, y ubica el resultado en ACCD. El acumulador A corresponde a los 8 bits altos del doble acumulador D de 16 bits.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	⇑

La operación ADDD es ADDD(opr)

1.13.7 **AND** Operación lógica AND.

Operación: ACCX ⇐ (ACCX) . (M)

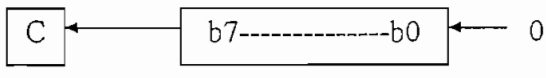
Descripción: Realiza la lógica AND entre los contenidos de ACCX y el contenido de M y ubica el resultado en ACCX. (Cada bit de ACCX después de la operación será la lógica AND de los bits correspondientes de M y ACCX antes de la operación).

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	-	-

La operación AND puede ser ANDA(opr) o ANDB(opr).

1.13.8 **ASL** Desplazamiento a la izquierda.

Operación: 

Descripción: desplaza todos los bits del ACCX o M un lugar a la izquierda. El bit cero es cargado con un cero. El bit C del CCR es cargado desde el bit más significativo del ACCX o M.

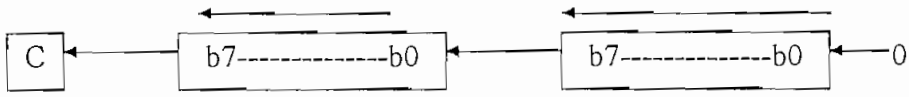
El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	⇑

La operación ASL puede ser ASLA(opr), ASLB(opr), ASL(opr).

1.13.9 ASLD Desplazamiento a la izquierda del doble acumulador.

Operación:



ACCA ACCB

Descripción: Desplaza todos los bits del acumulador D un lugar a la izquierda. El bit cero se carga con cero. El bit C del CCR es cargado con el bit más significativo del ACCD.

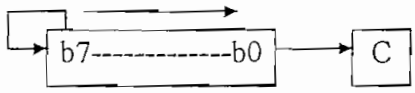
El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	⇑

La operación ASLD es: ASLD.

1.13.10 ASR Desplazamiento a la derecha.

Operación:



Descripción: Desplaza todo el contenido del ACCX o M un lugar a la derecha. El bit 7 se mantiene constante. El bit cero es cargado en el bit C del CCR.Esta

operación divide un valor en complemento de dos para dos, sin cambiar el signo. El bit del carry puede ser usado para redondiar el resultado.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	⇑

La operación ASR puede ser: ASRA, ASRB, ASR(opr).

1.13.11 BCC Salte si el carry es cero.

Operación: $PC \Leftarrow (PC) + \$0002 + Rel$ si $(C) = 0$

Descripción: Prueba el estado del bit C del CCR y causa un salto si C es cero.

El estado del no se altera.

La operación BCC es: BCC(rel).

1.13.12 BCLR Borra los bits en Memoria

Operación: $M \Leftarrow (M).\overline{(PC+2)}$

$M \Leftarrow (M).\overline{(PC+3)}$ (solo para direccionamiento indexado a Y).

Descripción: Borra múltiples bits de la localidad de memoria M. Los bits a ser borrados son especificados con uno en el byte de la máscara,todos los demás bits en la memoria M son reescritos a su estado actual.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	-	-

La operación BCLR es: BCLR(opr)(mask).

1.13.13 **BCS** Salte si el carry es uno.

Operación: PC ⇐ (PC) + $0002 + Rel si (C) = 1

Descripción: Prueba el estado del bit C del CCR y causa un salto si C es uno.

El estado del CCR no se altera.

La operación BCS es: BCS(rel).

1.13.14 **BEQ** Salte si es igual

Operación: PC ⇐ (PC) + $0002 + Rel si (Z) = 1

Descripción: Prueba el estado del bit Z del CCR y causa un salto si Z es uno.

El estado del CCR no se altera.

La operación BEQ es: BEQ(rel).

1.13.15 **BGE** Salte si es mayor o igual que cero

Operación: PC ⇐ (PC) + $0002 + Rel si (N) ⊕ (V) = 0

Por ejemplo: si (ACCX) ≥ (M)

Descripción: Si la instrucción BGE es ejecutada inmediatamente después de la ejecución de cualquiera de las instrucciones: CBA, CMP(A,B, o D), CP(XoY), SBA, SUB(A, B, 0 D), el salto ocurrirá si y solo si el número en complemento de dos representado por el ACCX fue mayor o igual al número en complemento de dos representado por M.

El estado del CCR no se altera.

La operación BGE es: BGE(rel).

1.13.16 **BGT** Salte si es mayor que cero

Operación: PC ⇐ (PC) + $0002 + Rel si (Z) + [(N) ⊕ (V)] = 0

Por ejemplo: si (ACCX) > (M)

Descripción: Si la instrucción BGT es ejecutada inmediatamente después de la ejecución de cualquiera de las instrucciones: CBA, CMP(A,B, o D), CP(XoY), SBA, SUB(A, B, 0 D), el salto ocurrirá si y solo si el número en complemento de dos representado por el ACCX fue mayor al número en complemento de dos representado por M.

El estado del CCR no se altera.

La operación BGT es: BGT(rel).

1.13.17 **BHI** Salte si es mayor (utilizado para números binarios sin signo)

Operación: PC ⇐ (PC) + $0002 + Rel si (C) + (Z) = 0

Por ejemplo: si (ACCX) > (M)

Descripción: Si la instrucción BHI es ejecutada inmediatamente después de la ejecución de cualquiera de las instrucciones: CBA, CMP(A,B, o D), CP(XoY), SBA, SUB(A, B, 0 D), el salto ocurrirá si y solo si el número binario sin signo representado por el ACCX fue mayor al número binario representado por M.

El estado del CCR no se altera.

La operación BHI es: BHI(rel).

1.13.18 **BHS** Salte si es mayor o igual (utilizado para números binarios sin signo)

Operación: PC ⇐ (PC) + $0002 + Rel si (C) = 0

Por ejemplo: si (ACCX) ≥ (M)

Descripción: Si la instrucción BHS es ejecutada inmediatamente después de la ejecución de cualquiera de las instrucciones: CBA, CMP(A,B, o D), CP(XoY), SBA, SUB(A, B, 0 D), el salto ocurrirá si y solo si el número binario sin signo representado por el ACCX fue mayor o igual al número binario representado por M.

El estado del CCR no se altera.

La operación BHS es: BHS(rel).

1.13.19 **BIT** Prueba del estado del bit

Operación: (ACCX).(M)

Descripción: Realiza la operación lógica AND entre el contenido de ACCX y el contenido de la memoria y modifica los bits N y Z del CCR. Ni el contenido de ACCX, ni el de M son afectados.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	-	-

La operación BIT puede ser: BITA(opr) o BITB(opr).

1.13.20 **BLE** Salte si es menor o igual que cero

Operación: PC ⇐ (PC) + $0002 + Rel si (Z) + [(N) ⊕ (V)] = 1

Por ejemplo: si (ACCX) ≤(M)

Descripción: Si la instrucción BLE es ejecutada inmediatamente después de la ejecución de cualquiera de las instrucciones: CBA, CMP(A,B, o D), CP(XoY), SBA, SUB(A, B, 0 D), el salto ocurrirá si y solo si el número en complemento de dos representado por el ACCX fue menor o igual al número en complemento de dos representado por M.

El estado del CCR no se altera.

La operación BLE es: BLE(rel).

1.13.21 **BLO** Salte si es menor (utilizado para números binarios sin signo)

Operación: PC ⇐ (PC) + $0002 + Rel si (C) = 1

Por ejemplo: si (ACCX) < (M)

Descripción: Si la instrucción BLO es ejecutada inmediatamente después de la ejecución de cualquiera de las instrucciones: CBA, CMP(A,B, o D), CP(XoY), SBA, SUB(A, B, 0 D), el salto ocurrirá si y solo si el número binario sin signo representado por el ACCX fue menor al número binario sin signo representado por M.

El estado del CCR no se altera.

La operación BLO es: BLO(rel).

1.13.22 **BLS** Salte si es menor o igual (utilizado para números binarios sin signo)

Operación: PC $\Leftarrow$ (PC) + \$0002 + Rel si (C) + (Z) = 1

Por ejemplo: si (ACCX) $\leq$ (M)

Descripción: Si la instrucción BLS es ejecutada inmediatamente después de la ejecución de cualquiera de las instrucciones: CBA, CMP(A,B, o D), CP(XoY), SBA, SUB(A, B, 0 D), el salto ocurrirá si y solo si el número binario sin signo representado por el ACCX fue menor o igual al número binario sin signo representado por M.

El estado del CCR no se altera.

La operación BLS es: BLS(rel).

1.13.23 **BLT** Salte si es menor que cero (usado para números en complemento de dos con signo)

Operación: PC $\Leftarrow$ (PC) + \$0002 + Rel si [(N) $\oplus$ (V)] = 1

Por ejemplo: si (ACCX) $\leq$(M)

Descripción: Si la instrucción BLT es ejecutada inmediatamente después de la ejecución de cualquiera de las instrucciones: CBA, CMP(A,B, o D), CP(XoY), SBA, SUB(A, B, 0 D), el salto ocurrirá si y solo si el número en complemento de dos representado por el ACCX fue menor o al número en complemento de dos representado por M.

El estado del CCR no se altera.

La operación BLT es: BLT(rel).

1.13.24 BMI Salte si es negativo

Operación: PC ⇐ (PC) + $0002 + Rel si (N) = 1

Descripción: Prueba el estado del bit N en el CCR y causa un salto si N es uno.

El estado del CCR no se altera.

La operación BMI es: BMI(rel).

1.13.25 BNE Salte si no es cero.

Operación: PC ⇐ (PC) + $0002 + Rel si (Z) = 0

Descripción: Prueba el estado del bit Z en el CCR y causa un salto si Z es cero.

El estado del CCR no se altera.

La operación BNE es: BNE(rel).

1.13.26 BPL Salte si no es negativo

Operación: PC ⇐ (PC) + $0002 + Rel si (N) = 0

Descripción: Prueba el estado del bit N en el CCR y causa un salto si N es cero.

El estado del CCR no se altera.

La operación BPL es: BPL(rel).

1.13.27 BRA Salte siempre

Operación: PC ⇐ (PC) + $0002 + Rel

Descripción: Salto incondicional a la dirección dada por la fórmula anterior, en la cuál Rel es el offset relativo almacenado como un número en

complemento de dos en el segundo byte del codigo de máquina correspondiente a la instrucción de salto.

El estado del CCR no se altera.

La operación BRA es: BRA(rel).

1.13.28 **BRCLR** Salte si el (los) bit(s) son cero

Operación: PC ⇐ (PC) + $0004 + Rel si (M).(PC+2) = 0

PC ⇐ (PC) + $0005 + Rel si (M).(PC+3) = 0 (para direccionamiento indexado con el registro Y)

Descripción: Realiza la lógica AND entre el contenido de M y la máscara dada con la instrucción y salta si el resultado es cero (solo si todos los bits correspondientes a unos en la máscara son ceros en el byte probado).

El estado del CCR no se altera.

S	X	H	I	N	Z	V	C
-	-	-	-	-	-	-	-

La operación BRCLR es: BRCLR(opr)(msk)(rel).

1.13.29 **BRN** Nunca salte

Operación: PC ⇐ (PC) + $0002

Descripción: Esta instrucción puede ser considerada como dos instrucciones NOP requiriendo tres ciclos para su ejecución. Su inclusión en el set de instrucciones es para dar una instrucción de complemento a la instrucción BRA.

El estado del CCR no se altera.

La operación BRN es: BRN(rel).

1.13.30 **BRSET** Salte si los bits son uno.

Operación: $PC \Leftarrow (PC) + \$0004 + Rel$ si $(M).(PC+2) = 0$

$PC \Leftarrow (PC) + \$0005 + Rel$ si $(M).(PC+3) = 0$ (para direccionamiento indexado con el registro Y)

Descripción: Realiza la lógica AND entre el contenido de M invertido y la máscara dada con la instrucción y salta si el resultado es cero (solo si todos los bits correspondientes a unos en la máscara son unos en el byte probado).

El estado del CCR no se altera.

La operación BRSET es: BRSET(opr)(msk)(rel).

1.13.31 BSET Setea los bits en memoria

Operación: $M \Leftarrow (M) + (PC+2)$

$M \Leftarrow (M) + (PC+3)$ (para direccionamiento indexado con el registro Y)

Descripción: Setea múltiples bits en la localidad M. El(los) bit(s) a ser seteado(s) son especificados por unos en el byte de la mascara, todos los demás bits en M no son afectados.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	-	-

La operación BSET es: BSET(opr)(msk)(rel).

1.13.32 BSR Salto a subrutina.

Operación: $PC \Leftarrow (PC) + \$0002$

$\Downarrow (PCL)$ Push el byte bajo del PC

$SP \Leftarrow (SP) - 0001$

⇓ (PCH) Push el byte alto del PC

SP ⇐ (SP) - 0001

PC ⇐ (PC) + Rel

Descripción: El contador de programa es incrementado en dos (ésta será la dirección de retorno). EL byte menos significativo del contador de programa es almacenado en el puntero de pila. El puntero de pila es decrementado en uno. El byte menos significativo del contador de programa es almacenado en el puntero de pila. El puntero de pila es decrementado en uno. Un salto entonces ocurre a la localidad especificada por el offset del salto.

El estado del CCR no se altera.

La operación BSR es: BSR(opr)(msk)(rel).

1.13.33 BVC Salte si no hay desbordamiento

Operación: PC ⇐ (PC) + $0002 si (V) = 0

Descripción: Prueba el estado del bit V en el CCR y causa un salto si V es cero.

El estado del CCR no se altera.

La operación BVC es: BVC(rel).

1.13.34 BVS Salte si hay desbordamiento

Operación: PC ⇐ (PC) + $0002 si (V) = 1

Descripción: Prueba el estado del bit V en el CCR y causa un salto si V es uno.

El estado del CCR no se altera.

La operación BVS es: BVS(rel).

1.13.35 CBA Comparación de acumuladores

Operación: (ACCA) - (ACCB)

Descripción: Compara el contenido del acumulador A con el contenido del acumulador B y varía el estado de los bits N, Z, V, C del CCR, los cuales pueden ser usados para saltos condicionales aritméticos y lógicos. Ninguno de los dos operandos es afectado.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	⇑

La operación CBA es: CBA.

1.13.36 CLC Limpiar el carry

Operación: C ⇐ 0

Descripción: Limpia el bit C en el CCR.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	-	-	-	0

La operación CLC es: CLC.

1.13.37 CLI Limpia la máscara de interrupción I

Operación: I ⇐ 0

Descripción: Limpia el bit I de la máscara de interrupción I en el CCR. Cuando el bit I es borrado, las interrupciones son habilitadas.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	0	-	-	-	-

La operación CLI es: CLI.

1.13.38 **CLR** Limpiar

Operación: ACCX $\Leftarrow$ 0 ó M $\Leftarrow$ 0

Descripción: Los contenidos de ACCX o de M son remplazados con ceros.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	0	1	0	0

La operación CLR puede ser: CLRA, CLRB, CLR(opr).

1.13.39 **CLV** Limpia el bit de desbordamiento en complemento de dos

Operación: V $\Leftarrow$ 0

Descripción: Limpia el bit V sobreflujo en complemento de dos en el CCR.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	-	-	0	-

La operación CLV es: CLVI

1.13.40 **CMP** Compare

Operación: (ACCA) - (M)

Descripción: Compara el contenido de ACCX con el contenido de M y varía el estado de los bits N, Z, V, C del CCR, los cuales pueden ser usados para saltos condicionales aritméticos y lógicos. Ninguno de los dos operandos es afectado.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	⇑

La operación CMP puede ser: CMPA(opr), CMPB(opr).

1.13.41 COM Complemento

Operación: $ACCX \Leftarrow (ACCX) = \$FF - (ACCX)$ ó $M \Leftarrow (M) = \$FF - (M)$

Descripción: cambia los contenidos de ACCX o M por su complemento (Cada bit del contenido de ACCX o M es reemplazado con el complemento de ése bit.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	0	1

La operación COM puede ser: COMA, COMB, COM(opr).

1.13.42 CPD Compare el doble acumulador

Operación: (ACCD) - (M:M+1)

Descripción: Compara los contenidos del acumulador D con el valor de 16 bits en la dirección especificada y varía el estado de los bits N, Z, V, C del CCR, los cuales pueden ser usados para saltos condicionales aritméticos y lógicos. Ninguno de los dos operandos es afectado.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	⇑

La operación CPD es: CPD(opr)

1.13.43 **CPX** Compare el registro índice X

Operación: (IX) - (M:M+1)

Descripción: Compara los contenidos del registro índice X con el valor de 16 bits en la dirección especificada y varía el estado de los bits N, Z, V, C del CCR, los cuales pueden ser usados para saltos condicionales aritméticos y lógicos. Ninguno de los dos operandos es afectado.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	⇑

La operación CPX es: CPX(opr)

1.13.44 **CPX** Compare el registro índice Y

Operación: (IY) - (M:M+1)

Descripción: Compara los contenidos del registro índice Y con el valor de 16 bits en la dirección especificada y varía el estado de los bits N, Z, V, C del CCR, los cuales pueden ser usados para saltos condicionales aritméticos y lógicos. Ninguno de los dos operandos es afectado.

El estado del CCR es:

S	X	H	I	N	Z	V	C

-	-	-	-	⇑	⇑	⇑	⇑

La operación CPX es: CPY(opr)

1.13.45 DAA Ajuste decimal del acumulador A

Descripción: Si el contenido del acumulador A, el estado del carry C y el estado del carry intermedio H son afectados después de aplicar una de las siguientes operaciones ABA, ADD, o ADC a operandos BCD con o sin un carry inicial, la operación DAA ajustará el contenido del acumulador A y el bit C (carry) en el CCR para representar el resultado de la suma en BCD (decimal codificado en binario).

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	?	⇑

? = no definido.

La operación DAA es: DAA

1.13.46 DEC Decremente

Operación: ACCX ⇐ (ACCX) - $01 ó M ⇐ (M) -$01

Descripción: Resta uno de los contenidos de ACCX o de M

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	-

La operación DEC puede ser: DECA, DECB, DEC(opr).

1.13.47 DES Decremente el Contador de Pila (Stack Pointer)

Operación: SP ⇐ (SP) - $0001

Descripción: Resta uno del contador de pila

El estado del CCR no se altera.

La operación DES es: DES:

1.13.48 DEX Decremente el registro índice X

Operación: IX ⇐ (IX) - $0001

Descripción: Resta uno del registro índice X. Solo el bit Z es seteado o borrado de acuerdo al resultado de ésta operación.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	-	⇑	-	-

La operación DEX es: DEX:

1.13.49 DEY Decremente el registro índice Y

Operación: IY ⇐ (IY) - $0001

Descripción: Resta uno del registro índice Y. Solo el bit Z es seteado o borrado de acuerdo al resultado de ésta operación.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	-	⇑	-	-

La operación DEY es: DEY:

1.13.50 EOR OR exclusivo

Operación: ACCX ⇐ (ACCX) ⊕ (M)

Descripción: Realiza la operación lógica OR exclusivo entre el contenido de ACCX y el contenido de M y ubica el resultado en ACCX. (Cada bit de ACCX después de la operación será igual a la operación lógica OR exclusivo entre los correspondientes bits de M y ACCX antes de la operación).

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	0	-

La operación EOR puede ser: EORA(opr) ó EORB(opr).

1.13.51 FDIV División fraccional

Operación: (ACCD)/(IX); IX⇐ cociente ACCD ⇐ residuo

Descripción: Realiza una división fraccional sin signo del numerador de 16 bits almacenado en el acumulador D para el denominador de 16 bits almacenado en el registro índice X y altera el estado de los bits Z, V y C en el CCR de acuerdo a los resultados. El cociente es puesto en el registro índice X, y el residuo es situado en el acumulador D. Se asume que el punto decimal está en el mismo lugar para el numerador y el denominador. El punto decimal para el cociente se ubica a la izquierda del bit 15. Se asume que el numerador es menor que el denominador, en el caso de que el numerador sea menor o igual que el numerador (overflow) o división por cero, el cociente se llena con $FFFF y el residuo es indeterminado.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	-	⇑	⇑	⇑

C=1 si denominador es 0.

La operación FDIV es: FDIV

1.13.52 IDIV División entera

Operación: (ACCD)/(IX); IX⇐ cociente ACCD ⇐ residuo

Descripción: Realiza una división entera sin signo del numerador de 16 bits almacenado en el acumulador D para el denominador de 16 bits almacenado en el registro índice X y altera el estado de los bits Z y C en el CCR de acuerdo a los resultados. El cociente es puesto en el registro índice X, y el residuo es situado en el acumulador D. Se asume que el punTo decimal está en el mismo lugar para el numerador y el denominador. El punto decimal para el cociente se ubica a la derecha del bit 0., en el caso de división por cero, el cociente se llena con $FFFF y el residuo es indeterminado.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	-	⇑	0	⇑

C=1 si denominador es 0.

La operación IDIV es: IDIV

1.13.53 INC Incremente

Operación: ACCX ⇐ (ACCX) + $01 ó M ⇐ (M) + $01

Descripción: Suma uno a los contenidos de ACCX o de M

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	-

La operación INC puede ser: INCA, INCB, INC(opr).

1.13.54 INS Incremente el Contador de Pila (Stack Pointer)

Operación: SP ⇐ (SP) + $0001

Descripción: suma uno al contador de pila

El estado del CCR no se altera.

La operación INS es: INS:

1.13.55 INX Incremente el registro índice X

Operación: IX ⇐ (IX) + $0001

Descripción: suma uno al registro índice X. Solo el bit Z es seteado o borrado de acuerdo al resultado de ésta operación.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	-	⇑	-	-

La operación INX es: INX:

1.13.56 INY Incremente el registro índice Y

Operación: IY ⇐ (IY) + $0001

Descripción: suma uno al registro índice Y. Solo el bit Z es seteado o borrado de acuerdo al resultado de ésta operación.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	-	⇑	-	-

La operación INY es: INY:

1.13.57 JMP Salto

Operación: PC ⇐ dirección efectiva

Descripción: Un salto ocurre a la instrucción almacenada en la dirección efectiva. La dirección efectiva es obtenida de acuerdo a las re4glas del direccionamiento extendido o indexado.

El estado del CCR no se altera.

La operación JMP es: JMP(opr).

1.13.58 JSR Salto a subrutina

Operación: PC ⇐ (PC) + $0003 Para direccionamiento extendido o indexado a Y

PC ⇐ (PC) + $0002 Para direccionamiento directo o indexado a X

⇓ (PCL) Push el byte bajo del PC

SP ⇐ (SP) - 0001

⇓ (PCH) Push el byte alto del PC

SP ⇐ (SP) - 0001

PC ⇐ Dirección efectiva

Descripción: El contador de programa es incrementado en tres o dos, dependieendo del modo de direccionamiento, y es entonces puesto en el puntero de pila, ocho bits a la vez, primero el byte menos significativo. Un salto ocurre a la dirección efectiva. La dirección efectiva es obtenida de acuerdo a las reglas del direccionamiento directo, extendido o indexado.

El estado del CCR no se altera.

La operación JSR es: JSR(opr).

1.13.59 **LDA** Cargar el acumulador

Operación: ACCX ⇐ (M)

Descripción: Carga el contenido de la memoria M en el acumulador de 8 bits. Los bits del CCR son alterados de acuerdo al dato.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	-	-

La operación LDA puede ser: LDAA(opr) ó LDAB(opr).

1.13.61 **LDD** Cargar el doble acumulador

Operación: ACCD ⇐ (M:M+1); ACCA ⇐ (M) y ACCB ⇐ (M+1)

Descripción: Carga el contenido de las localidades de memoria M y M+1 en el doble acumulador. Los bits del CCR son alterados de acuerdo al dato. La información de la localidad M es almacenada en el ACCA y la de M+1 en ACCB.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	-	-

La operación LDD es: LDD(opr).

1.13.61 **LDS** Cargar al Puntero de pila (stack pointer).

Operación: SPH ⇐ (M), SPL ⇐ (M+1)

Descripción: Carga al byte más significativo del puntero el contenido de M y al byte menos significativo del puntero el contenido de M+1.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	-	-

La operación LDD es: LDS(opr).

1.13.62 **LDX** Carga al registro índice X.

Operación: IXH ⇐ (M), IXL ⇐ (M+1)

Descripción: Carga al byte más significativo del registro índice X el contenido de M y al byte menos significativo del registro índice X el contenido de M+1.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	-	-

La operación LDX es: LDX(opr).

1.13.63 **LDY** Cargar al registro índice Y

Operación: IYH ⇐ (M), IYL ⇐ (M+1)

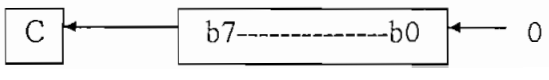
Descripción: Carga al byte más significativo del registro índice Y el contenido de M y al byte menos significativo del registro índice Y el contenido de M+1.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	-	-

La operación LDY es: LDY(opr).

1.13.64 **LSL** desplazamiento lógico a izquierda

Operación : 

Descripción: Rota todos los bits del ACCX o de M un lugar a la izquierda. El bit cero es cargado con cero. El bit C es cargado del bit mas significativo de ACCX o M.

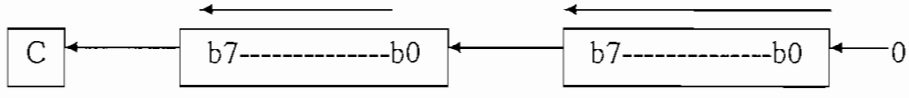
El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	⇑

La operación LSL puede ser: LSLA, LSLB, LSL(opr).

1.13.65 LSLD Desplazamiento lógico a izquierda del doble acumulador.

Operación:



ACCA ACCB

Descripción: Desplaza todos los bits del acumulador D un lugar a la izquierda. El bit cero se carga con cero. El bit C del CCR es cargado con el bit más significativo del ACCD.

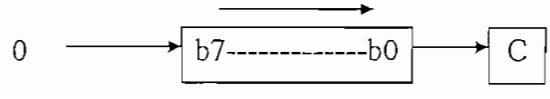
El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	⇑

La operación LSLD es: LSLD.

1.13.66 LSR Desplazamiento lógico a derecha.

Operación:



Descripción: Desplaza todo el contenido del ACCX o M un lugar a la derecha. El bit 7 es cargado con cero. El bit C es cargado del bit menos significativo de ACCX.

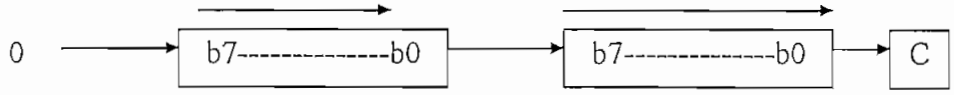
El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	0	⇑	⇑	⇑

La operación LSR puede ser: LSRA, LSRB, LSR(opr).

1.13.67 LSRD Rotación lógica a derecha del doble acumulador.

Operación:



ACCA ACCB

Descripción: Desplaza todos los bits del acumulador D un lugar a la derecha. El bit 15 se carga con cero. El bit C del CCR es cargado con el bit menos significativo del ACCD.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	0	⇑	⇑	⇑

La operación LSRD es: LSRD.

1.13.68 MUL Multiplicación sin signo.

Operación: ACCD ⇐ (ACCA) . (ACCB)

Descripción: Multiplica el valor binario sin signo del acumulador A por el valor binario sin signo del acumulador B para obtener un resultado de 16 bits sin signo en el doble acumulador D. La bandera del carry permite redondear el byte mas significativo del resultado a travez de la secuencia: MUL, ADCA #0.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	-	-	-	⇑

La operación MUL es: MUL.

1.13.69 **NEG** Sacar el negativo

Operación: (ACCX) ⇐ -(ACCX)= $00 - (ACCX) ó (M) ⇐ -(M) =$00 - (M)

Descripción: Reemplaza los contenidos de ACCX o M por su complemento de dos. EL valor $80 no cambia.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	⇑

La operación NEG puede ser: NEGA, NEGB o NEG(opr).

1.13.70 **NOP** No operación.

Descripción: ésta es una instrucción de un solo bit que causa únicamente que el contador de programa sea incrementado en uno. Ningún otro registro es afectado.

El estado del CCR no se altera.

La operación NOP es NOP

1.13.71 **OR** Operación lógica OR inclusivo.

Operación: ACCX ⇐ (ACCX) + (M)

Descripción: Realiza la lógica OR inclusivo entre el contenido del ACCX y el contenido de M y ubica el resultado en ACCX.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	0	-

La operación ORA puede ser: ORAA(opr), ORAB(opr).

1.13.72 **PSH** Poner el dato en el puntero (stack)

Operación: ⇓ACCX, SP ⇐ (SP) - $0001

Descripción: El contenido del ACCX es almacenado en la dirección contenida en el puntero de pila. El puntero de pila es entonces decrementado. Esta instrucción es comunmente usada para guardar el contenido de uno o más registros del CPU al inicio de una subrutina. Solo antes de retornar de la subrutina, la correspondiente instrucción PULL es usada para restablecer el valor almacenado de los registros del CPU.

El estado del CCR no se altera.

La operación PSH puede ser: PSHA, PSHB.

1.13.73 **PSHX** Poner el registro índice X en el puntero (stack)

Operación: ⇓IXL, SP ⇐ (SP) - $0001

⇓IXH, SP ⇐ (SP) - $0001

Descripción: El contenido del registro índice X es almacenado en la dirección contenida en el puntero de pila. El puntero de pila es entonces decrementado en dos.

Esta instrucción es comunmente usada para guardar el contenido de uno o más registros del CPU al inicio de una subrutina. Solo antes de retornar de la subrutina, la correspondiente instrucción PULL es usada para restablecer el valor almacenado de los registros del CPU.

El estado del CCR no se altera.

La operación PSHX es: PSHX.

1.13.74 PSHY Poner el registro índice Y en el puntero (stack)

Operación: ⇓IYL, SP ⇐ (SP) - $0001

⇓IYH, SP ⇐ (SP) - $0001

Descripción: El contenido del registro índice Y es almacenado en la dirección contenida en el puntero de pila. El puntero de pila es entonces decrementado en dos. Esta instrucción es comunmente usada para guardar el contenido de uno o más registros del CPU al inicio de una subrutina. Solo antes de retornar de la subrutina, la correspondiente instrucción PULL es usada para restablecer el valor almacenado de los registros del CPU.

El estado del CCR no se altera.

La operación PSHY es: PSHY.

1.13.75 PUL Sacar el dato del Puntero (Stack)

Operación: SP ⇐ (SP) + $0001, ⇑(ACCX)

Descripción: El puntero de pila es incrementado. El ACCX es entonces cargado del puntero de pila. La instrucción PSH es comunmente usada para guardar el contenido de uno o más registros del CPU al inicio de una subrutina. Solo antes de retornar de la subrutina, la correspondiente instrucción PUL es usada para restablecer el valor almacenado de los registros del CPU.

El estado del CCR no se altera.

La operación PUL puede ser: PULA, PULB.

1.13.76 **PULX** Sacar el registro índice X del Puntero (Stack)

Operación: SP ⇐ (SP) + $0001, Î(IXH)

SP ⇐ (SP) + $0001, Î(IXL)

Descripción: El registro índice X es cargado del puntero de pila. El puntero de pila es incrementado en dos. La instrucción PSH es comunmente usada para guardar el contenido de uno o más registros del CPU al inicio de una subrutina. Solo antes de retornar de la subrutina, la correspondiente instrucción PUL es usada para restablecer el valor almacenado de los registros del CPU.

El estado del CCR no se altera.

La operación PULX es: PULX

1.13.77 **PULY** Sacar el registro índice Y del Puntero (Stack)

Operación: SP ⇐ (SP) + $0001, Î(IYH)

SP ⇐ (SP) + $0001, Î(IYL)

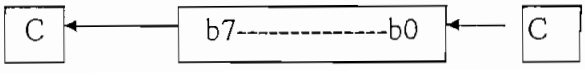
Descripción: El registro índice Y es cargado del puntero de pila. El puntero de pila es incrementado en dos. La instrucción PSH es comunmente usada para guardar el contenido de uno o más registros del CPU al inicio de una subrutina. Solo antes de retornar de la subrutina, la correspondiente instrucción PUL es usada para restablecer el valor almacenado de los registros del CPU.

El estado del CCR no se altera.

La operación PULY es: PULY

1.13.78 **ROL** Rotación a la izquierda

Operación:



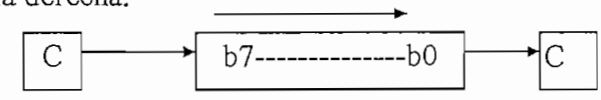
Descripción: Rota todos los bits del ACCX o de M un lugar a la izquierda. El bit cero es cargado del bit C. El bit C es cargado del bit más significativo del ACCX o de M.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	⇑

La operación ROL puede ser: ROLA, ROLB, ROL(opr).

1.13.79 **ROR** Rotación a la derecha.

Operación: 

Descripción: Rota todos los bits del ACCX o de M un lugar a la derecha. El bit 7 es cargado del bit C. El bit C es cargado del bit menos significativo del ACCX o de M.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	⇑

La operación ROL puede ser: RORA, RORB, ROR(opr).

1.13.80 **RTI** Retorno de interrupción.

Operación: $SP \Leftarrow (SP) + \$0001$, ⇑(CCR)

$SP \Leftarrow (SP) + \$0001$, ⇑(ACCB)

$SP \Leftarrow (SP) + \$0001$, ⇑(ACCA)

$SP \Leftarrow (SP) + \$0001$, ⇑(IXH)

$SP \Leftarrow (SP) + \$0001$, ⇑(IXL)

$SP \Leftarrow (SP) + \$0001, \quad \Uparrow(IYH)$

$SP \Leftarrow (SP) + \$0001, \quad \Uparrow(IYL)$

$SP \Leftarrow (SP) + \$0001, \quad \Uparrow(PCH)$

$SP \Leftarrow (SP) + \$0001, \quad \Uparrow(PCL)$

Descripción: El código de condiciones, los acumuladores B y A, los registros índices X, Y, y el contador de programa serán restablecidos al estado que tenían antes de ingresar a la interrupción. El bit X en el CCR podría ser borrado como un resultado de una instrucción RTI, pero no podría ser seteado si éste fue borrado antes de la ejecución de la instrucción RTI.

El estado del CCR es:

S	X	H	I	N	Z	V	C
⇑	⇑	⇑	⇓	⇑	⇑	⇑	⇑

La operación RTI es: RTI.

1.13.81 **RTS** Retorno de subrutina

Operación. $SP \Leftarrow (SP) + \$0001, \quad \Uparrow(PCH)$

$SP \Leftarrow (SP) + \$0001, \quad \Uparrow(PCL)$

Descripción: El puntero de pila es incrementado en uno. Los contenidos del byte de memoria de la dirección contenida en el puntero de pila son cargados en los ocho bits altos del contador de programa. El puntero de pila es otra vez incrementado en uno, y los contenidos del byte de memoria ahora contenidos en el puntero de pila son cargados en los ocho bits bajos del contador de programa.

El estado del CCR no se altera.

La operación RTS es: RTS.

1.13.82 SBA Resta de acumuladores

Operación: ACCA ⇐ (ACCA) - (ACCB)

Descripción: Resta el contenido del acumulador B del contenido del acumulador A y ubica el resultado en el acumulador A. El contenido del acumulador B no es afectado. Para instrucciones de resta, el bit C en el CCR representa un préstamo (llevo).

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	⇑

La operación SBA es: SBA.

1.13.83 SBC Resta con "llevo"

Operación: ACCX ⇐ (ACCX) - (M) - (C)

Descripción: Resta el contenido de memoria y el contenido del bit C del contenido de ACCX y ubica el resultado en el ACCX. Para instrucciones de resta, el bit C en el CCR representa un préstamo (llevo).

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	⇑

La operación SBC puede ser: SBCA(opr) ó SBCB(opr).

1.13.84 SEC Poner 1 en el carry

Operación: C ⇐ 1

Descripción: Pone 1 en el bit C del CCR.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	-	-	-	1

La operación SEC es: SEC.

1.13.85 SEI Poner 1 en la máscara de interrupción I

Operación: $I \Leftarrow 1$

Descripción: Pone 1 en la máscara de interrupción I en el CCR. Cuando se pone un uno el el bit I todas las interrupciones enmascarables son inhibidas, y el microcontrolador reconocera solo las fuentes de interrupción no enmascarables y la interrupción SWI.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	1	-	-	-	-

La operación SEI es: SEI.

1.13.86 SEV Poner 1 en el bit de desbordamiento en complemento de dos.

Operación: $V \Leftarrow 1$

Descripción: Pone 1 en el bit V (desbordamiento en complemento de dos) del CCR.

El estado del CCR es:

S	X	H	I	N	Z	V	C

-	-	-	-	-	-	1	-

La operación SEV es: SEV.

1.13.87 STA Almacenar el acumulador

Operación: M ⇐ (ACCX)

Descripción: Almacena el contenido de ACCX en memoria. El contenido de ACCX no cambia.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	0	-

La operación STA puede ser: STAA(opr) ó STAB(opr).

1.13.88 STD Almacenar el doble acumulador

Operación: M: M+1 ⇐ (ACCD); M ⇐ (ACCA), M+1 ⇐ (ACCB)

Descripción: Almacena el contenido del doble acumulador ACCD en memoria. El contenido de ACCD no cambia.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	0	-

La operación STD es: STD(opr)

1.13.89 STOP Proceso de parada

Descripción: Si el bit S en el CCR es uno, entonces la instrucción STOP y la operación NOP son deshabilitadas. Si el bit S en el CCR es cero, la instrucción STOP

causa que todo el sistema se detenga y se mantenga en modo de stanby con consumo mínimo de energía. Todos los registros del CPU así como los pines de entrada salida se mantienen invariantes. Para regresar a trabajo normal se lo consigue a través de un RESET, o por las interrupciones XIRQ o IRQ.

El estado del CCR no se altera.

La operación STOP es: STOP

1.13.90 STS Almacene el puntero de pila (Stack pointer)

Operación: M: M+1 ⇐ (SP); M ⇐ (SPH), M+1 ⇐ (SPL)

Descripción: Almacena el contenido del puntero de pila en memoria. El contenido del puntero de pila no cambia.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	0	-

La operación STS es: STS(opr)

1.13.91 STX Almacene el registro índice X

Operación: M: M+1 ⇐ (IX); M ⇐ (IXH), M+1 ⇐ (IXL)

Descripción: Almacena el contenido del registro índice X en memoria. El contenido del registro índice X no cambia.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	0	-

La operación STX es: STX(opr)

1.13.92 **STY** Almacene el registro índice Y

Operación: M: M+1 ⇐ (IY); M ⇐ (IYH), M+1 ⇐ (IYL)

Descripción: Almacena el contenido del registro índice Y en memoria. El contenido del registro índice Y no cambia.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	0	-

La operación STY es: STY(opr)

1.13.93 **SUB** Resta.

Operación: ACCX ⇐ (ACCX) - (M)

Descripción: Resta el contenido de memoria de del contenido del acumulador X y ubica el resultado en ACCX. Para instrucciones de resta, el bit C en el CCR representa un "llevo".

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	⇑

La operación SUB puede ser: SUBA(opr) o SUBB(opr).

1.13.94 **SUBD** Resta al doble acumulador.

Operación: ACCD ⇐ (ACCD) - (M:M+1)

Descripción: Resta los contenidos de las localidades de memoria M:M+1 del contenido del acumulador D y ubica el resultado en ACCD. Para instrucciones de resta, el bit C en el CCR representa un "llevo".

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	⇑	⇑

La operación SUB es: SUBD(opr).

1.13.95 **SWI** Interrupción de Software.

Operación: PC ⇐ (PC) + $0001

⇓(PCL), SP ⇐ (SP) - $0001

⇓(PCH), SP ⇐ (SP) - $0001

⇓(IYL), SP ⇐ (SP) - $0001

⇓(IYH), SP ⇐ (SP) - $0001

⇓(IXL), SP ⇐ (SP) - $0001

⇓(IXH), SP ⇐ (SP) - $0001

⇓(ACCA), SP ⇐ (SP) - $0001

⇓(ACCB), SP ⇐ (SP) - $0001,

⇓(CCR), SP ⇐ (SP) - $0001

I ⇐ 1, PC ⇐ (vector SWI)

Descripción: El contador de programa es incrementado en uno. El contador de programa, los registros índice Y y X, los acumuladores A, B y el CCR son puestos en el puntero de pila. Luego el bit I en el CCR es seteado. El contador de programa es cargado con la dirección almacenada en el vector SWI, y ejecuta las instrucciones que se encuentran en las tres localidades de vector. Esta instrucción no es enmascarable por el bit I.

El estado del CCR no se altera.

La operación SWI es: SWI.

1.13.96 TAB Transferir del acumulador A al acumulador B

Operación: ACCB ⇐ (ACCA)

Descripción: Mueve el contenido del acumulador A al acumulador B. El contenido del acumulador A no es afectado.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	0	-

La operación TAB es: TAB

1.13.97 TAP Transferir del acumulador A al CCR

Operación: CCR ⇐ (ACCA)

Descripción: Transfiere el contenido de los bits 7 a 0 del acumulador A a las posiciones de los bits correspondientes en el CCR. El contenido del acumulador A no es afectado. EL bit X en el CCR puede ser borrado como resultado de una instrucción TAP pero no puede ser seteado si estubo borrado antes de la ejecución de ésta instrucción.

El estado del CCR es:

S	X	H	I	N	Z	V	C
⇑	⇓	⇑	⇑	⇑	⇑	⇑	⇑

La operación TAP es: TAP

1.13.98 TBA Transferir del acumulador B al acumulador A

Operación: ACCA ⇐ (ACCB)

Descripción: Mueve el contenido del acumulador B al acumulador A. El contenido del acumulador B no es afectado.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	0	-

La operación TBA es: TBA

1.13.99 TEST Operación de Prueba (Funciona solo en el modo de prueba).

Descripción: Esta es una instrucción de un solo byte que hace que el contador de programa sea incrementado continuamente. Esta instrucción solo puede ser utilizada en el modo de prueba. Para salir de ésta instrucción se debe resetear el CPU. El código de ésta instrucción es ilegal cuando no se está en el modo de prueba.

El estado del CCR no se altera.

La operación TEST es: TEST.

1.13.100 TPA Transferir del CCR al acumulador A

Operación: ACCA ⇐ (CCR)

Descripción: Transfiere el contenido de los bits 7 a 0 del CCR (registro de código de condiciones) a las posiciones de los bits correspondientes en el acumulador A.

El contenido del CCR no se altera.

La operación TPA es: TPA.

1.13.101 TST Prueba

Operación: (ACCX) - $00 ó (M) - $00

Descripción: Resta $00 del contenido de ACCX o M y setea el registro de condiciones de código. No se modifica ni el ACCX ni la localidad de memoria M. La información TST da una información mínima cuando prueba un valor sin signo, pero cuando un valor con signo es probado, todas las instrucciones de salto referentes al signo son utilizadas.

El estado del CCR es:

S	X	H	I	N	Z	V	C
-	-	-	-	⇑	⇑	0	0

La operación TST puede ser: TSTA, TSTB o TST(opr)

1.13.102 TSX Transferir del puntero de pila al registro índice X.

Operación: IX ⇐ (SP) + $0001

Descripción: Carga el registro índice X con el contenido del puntero de pila (stack pointer). El contenido del puntero de pila no cambia. Después de una instrucción TSX el registro índice X apunta al último valor que fue almacenado en la pila (stack).

El contenido del CCR no se altera.

La operación TSX es: TSX.

1.13.103 TSY Transferir del puntero de pila al registro índice Y.

Operación: IY ⇐ (SP) + $0001

Descripción: Carga el registro índice Y con el contenido del puntero de pila (stack pointer). El contenido del puntero de pila no cambia. Después de una instrucción TSY, el registro índice Y apunta al último valor que fue almacenado en la pila (stack).

El contenido del CCR no se altera.

La operación TSY es: TSY.

1.13.104 **TXS** Transferir del registro índice X al puntero de pila

Operación: SP $\Leftarrow$ (IX) - $0001

Descripción: Carga el puntero de pila con el contenido del registro índice X menos uno. El contenido del registro índice Y.

El contenido del CCR no se altera.

La operación TXS es: TXS.

1.13.105 **TSY** Transferir del registro índice Y al puntero de pila

Operación: SP $\Leftarrow$ (IY) - $0001

Descripción: Carga el puntero de pila con el contenido del registro índice Y menos uno. El contenido del registro índice Y.

El contenido del CCR no se altera.

La operación TXS es: TXS.

1.13.106 **WAI** Esperar por interrupción.

Operación: PC $\Leftarrow$ (PC) + $0001

$\Downarrow$(PCL), SP $\Leftarrow$ (SP) - $0001

$\Downarrow$(PCH), SP $\Leftarrow$ (SP) - $0001

$\Downarrow$(IYL), SP $\Leftarrow$ (SP) - $0001

$\Downarrow$(IYH), SP $\Leftarrow$ (SP) - $0001

$\Downarrow$(IXL), SP $\Leftarrow$ (SP) - $0001

$\Downarrow$(IXH), SP $\Leftarrow$ (SP) - $0001

$\Downarrow$(ACCA), SP $\Leftarrow$ (SP) - $0001

$\Downarrow$(ACCB), SP $\Leftarrow$ (SP) - $0001,

$\Downarrow$(CCR), SP $\Leftarrow$ (SP) - $0001

Descripción: El contador de programa es incrementado en uno. El contador de programa, los registros índice Y y X, los acumuladores A y B, y el CCR son puestos en la pila (stack). EL puntero de pila es decrementado en uno despues de que cada byte de dato es almacenado en la pila (stack). El MCU entoncesw ingresa en estado de espera por un número de ciclos de reloj. Mientras el MCU está en estado de espera, el bus de direcciones y datos repetidamente realiza lecturas de la dirección donde el CCR fue almacenado. EL MCU sale del modo de espera cuando una interrupción que no ha sido enmascarada ha sido sentida.

El contenido del CCR no se altera.

La operación WAI es: WAI.

1.13.107 **XGDX** Intercambiar entre el doble acumulador y el registro índice X

Operación: (IX) $\Longleftrightarrow$ (ACCD)

Descripción: Intercambia los contenidos del doble acumulador ACCD y el contenido del registro índice X.

El contenido del CCR no se altera.

La operación XGDX es: XGDX.

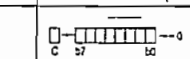
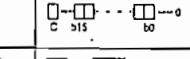
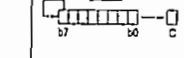
1.13.108 **XGDY** Intercambiar entre el doble acumulador y el registro índice Y

Operación: (IY) $\Longleftrightarrow$ (ACCD)

Descripción: Intercambia los contenidos del doble acumulador ACCD y el contenido del registro índice Y.

El contenido del CCR no se altera.

Table 10-1. MC68HC11A8 Instructions, Addressing Modes, and Execution Times (Sheet 1 of 7)

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal) Opcode	Operand(s)	Bytes	Cycle	Cycle by Cycle*	Condition Codes S X H I N Z V C
ABA	Add Accumulators	A + B → A	INH	1B		1	2	2-1	• • ↕ • ↕ ↕ ↕ ↕
ABX	Add B to X	IX + 00:B → IX	INH	3A		1	3	2-2	• • • • • • • •
ABY	Add B to Y	IY + 00:B → IY	INH	18 3A		2	4	2-4	• • • • • • • •
ADCA(opr)	Add with Carry to A	A + M + C → A	A IMM	89	ii	2	2	3-1	• • ↕ • ↕ ↕ ↕ ↕
			A DIR	99	dd	2	3	4-1	
			A EXT	B9	hh ll	3	4	5-2	
			A IND,X	A9	ff	2	4	6-2	
			A IND,Y	18 A9	ff	3	5	7-2	
ADCB (opr)	Add with Carry to B	B + M + C → B	B IMM	C9	ii	2	2	3-1	• • ↕ • ↕ ↕ ↕ ↕
			B DIR	D9	dd	2	3	4-1	
			B EXT	F9	hh ll	3	4	5-2	
			B IND,X	E9	ff	2	4	6-2	
			B IND,Y	18 E9	ff	3	5	7-2	
ADDA (opr)	Add Memory to A	A + M → A	A IMM	8B	ii	2	2	3-1	• • ↕ • ↕ ↕ ↕ ↕
			A DIR	9B	dd	2	3	4-1	
			A EXT	BB	hh ll	3	4	5-2	
			A IND,X	AB	ff	2	4	6-2	
			A IND,Y	18 AB	ff	3	5	7-2	
ADDB(opr)	Add Memory to B	B + M → B	B IMM	CB	ii	2	2	3-1	• • ↕ • ↕ ↕ ↕ ↕
			B DIR	DB	dd	2	3	4-1	
			B EXT	FB	hh ll	3	4	5-2	
			B IND,X	EB	ff	2	4	6-2	
			B IND,Y	18 EB	ff	3	5	7-2	
ADDD (opr)	Add 16-Bit to D	D + M:M + 1 → D	IMM	C3	jj kk	3	4	3-3	• • • • ↕ ↕ ↕ ↕
			DIR	D3	dd	2	5	4-7	
			EXT	F3	hh ll	3	6	5-10	
			IND,X	E3	ff	2	6	6-10	
			IND,Y	18 E3	ff	3	7	7-8	
ANDA (opr)	AND A with Memory	A•M → A	A IMM	84	ii	2	2	3-1	• • • • ↕ ↕ 0 •
			A DIR	94	dd	2	3	4-1	
			A EXT	B4	hh ll	3	4	5-2	
			A IND,X	A4	ff	2	4	6-2	
			A IND,Y	18 A4	ff	3	5	7-2	
ANDB (opr)	AND B with Memory	B•M → B	B IMM	C4	ii	2	2	3-1	• • • • ↕ ↕ 0 •
			B DIR	D4	dd	2	3	4-1	
			B EXT	F4	hh ll	3	4	5-2	
			B IND,X	E4	ff	2	4	6-2	
			B IND,Y	18 E4	ff	3	5	7-2	
ASL (opr)	Arithmetic Shift Left		EXT	78	hh ll	3	6	5-8	• • • • ↕ ↕ ↕ ↕
			IND,X	68	ff	2	6	6-3	
			IND,Y	18 68	ff	3	7	7-3	
ASLA			A INH	48		1	2	2-1	
ASLB			B INH	58		1	2	2-1	
ASLD	Arithmetic Shift Left Double		INH	05		1	3	2-2	• • • • ↕ ↕ ↕ ↕
ASR (opr)	Arithmetic Shift Right		EXT	77	hh ll	3	6	5-8	• • • • ↕ ↕ ↕ ↕
			IND,X	67	ff	2	6	6-3	
			IND,Y	18 67	ff	3	7	7-3	
ASRA			A INH	47		1	2	2-1	
ASRB			B INH	57		1	2	2-1	
BCC (rel)	Branch if Carry Clear	? C = 0	REL	24	rr	2	3	8-1	• • • • • • • •
BCLR (opr) (msk)	Clear Bit(s)	M•($\overline{mm}$) → M	DIR	15	dd mm	3	6	4-10	• • • • ↕ ↕ 0 •
			IND,X	1D	ff mm	3	7	6-13	
			IND,Y	18 1D	ff mm	4	8	7-10	
BCS (rel)	Branch if Carry Set	? C = 1	REL	25	rr	2	3	8-1	• • • • • • • •
BEQ (rel)	Branch if = Zero	? Z = 1	REL	27	rr	2	3	8-1	• • • • • • • •

*Cycle-by-cycle number provides a reference to Tables 10-2 through 10-8 which detail cycle-by-cycle operation.
Example: Table 10-1 Cycle-by-Cycle column reference number 2-4 equals Table 10-2 line item 2-4.

Table 10-1. MC68HC11A8 Instructions, Addressing Modes, and Execution Times (Sheet 2 of 7)

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal) Opcode	Operand(s)	Bytes	Cycle	Cycle by Cycle*	Condition Codes S X H I N Z V C
BGE (rel)	Branch if ≥ Zero	? N ⊕ V = 0	REL	2C	rr	2	3	8-1	- - - - - - - -
BGT (rel)	Branch if > Zero	? Z + (N ⊕ V) = 0	REL	2E	rr	2	3	8-1	- - - - - - - -
BHI (rel)	Branch if Higher	? C + Z = 0	REL	22	rr	2	3	8-1	- - - - - - - -
BHS (rel)	Branch if Higher or Same	? C = 0	REL	24	rr	2	3	8-1	- - - - - - - -
BITA (opr)	Bit(s) Test A with Memory	A•M	A IMM	85	ii	2	2	3-1	- - - - ↕ ↕ 0 -
			A DIR	95	dd	2	3	4-1	
			A EXT	B5	hh ll	3	4	5-2	
			A IND,X	A5	ff	2	4	5-2	
			A IND,Y	18 A5	ff	3	5	7-2	
BITB (opr)	Bit(s) Test B with Memory	B•M	B IMM	C5	ii	2	2	3-1	- - - - ↕ ↕ 0 -
			B DIR	D5	dd	2	3	4-1	
			B EXT	F5	hh ll	3	4	5-2	
			B IND,X	E5	ff	2	4	6-2	
			B IND,Y	18 E5	ff	3	5	7-2	
BLE (rel)	Branch if ≤ Zero	? Z + (N ⊕ V) = 1	REL	2F	rr	2	3	8-1	- - - - - - - -
BLO (rel)	Branch if Lower	? C = 1	REL	25	rr	2	3	8-1	- - - - - - - -
BLS (rel)	Branch if Lower or Same	? C + Z = 1	REL	23	rr	2	3	8-1	- - - - - - - -
BLT (rel)	Branch if < Zero	? N ⊕ V = 1	REL	2D	rr	2	3	8-1	- - - - - - - -
BMI (rel)	Branch if Minus	? N = 1	REL	2B	rr	2	3	8-1	- - - - - - - -
BNE (rel)	Branch if Not = Zero	? Z = 0	REL	26	rr	2	3	8-1	- - - - - - - -
BPL (rel)	Branch if Plus	? N = 0	REL	2A	rr	2	3	8-1	- - - - - - - -
BRA (rel)	Branch Always	? 1 = 1	REL	20	rr	2	3	8-1	- - - - - - - -
BRCLR(opr) (msk) (rel)	Branch if Bit(s) Clear	? M • mm = 0	DIR	13	dd mm rr	4	6	4-11	- - - - - - - -
			IND,X	1F	ff mm rr	4	7	6-14	
			IND,Y	18 1F	ff mm rr	5	8	7-11	
BRN (rel)	Branch Never	? 1 = 0	REL	21	rr	2	3	8-1	- - - - - - - -
BRSET(opr) (msk) (rel)	Branch if Bit(s) Set	? (M) • mm = 0	DIR	12	dd mm rr	4	6	4-11	- - - - - - - -
			IND,X	1E	ff mm rr	4	7	6-14	
			IND,Y	18 1E	ff mm rr	5	8	7-11	
BSET(opr) (msk)	Set Bit(s)	M + mm → M	DIR	14	dd mm	3	6	4-10	- - - - ↕ ↕ 0 -
			IND,X	1C	ff mm	3	7	6-13	
			IND,Y	18 1C	ff mm	4	8	7-10	
BSR (rel)	Branch to Subroutine	See Special Ops	REL	8D	rr	2	6	8-2	- - - - - - - -
BVC (rel)	Branch if Overflow Clear	? V = 0	REL	28	rr	2	3	8-1	- - - - - - - -
BVS (rel)	Branch if Overflow Set	? V = 1	REL	29	rr	2	3	8-1	- - - - - - - -
CBA	Compare A to B	A − B	INH	11		1	2	2-1	- - - - ↕ ↕ ↕ ↕
CLC	Clear Carry Bit	0 → C	INH	0C		1	2	2-1	- - - - - - - 0
CLI	Clear Interrupt Mask	0 → I	INH	0E		1	2	2-1	- - - 0 - - - -
CLR (opr)	Clear Memory Byte	0 → M	EXT	7F	hh ll	3	6	5-8	- - - - 0 1 0 0
			IND,X	6F	ff	2	6	6-3	
			IND,Y	18 6F	ff	3	7	7-3	
CLRA	Clear Accumulator A	0 → A	A INH	4F		1	2	2-1	- - - - 0 1 0 0
CLRB	Clear Accumulator B	0 → B	B INH	5F		1	2	2-1	- - - - 0 1 0 0
CLV	Clear Overflow Flag	0 → V	INH	0A		1	2	2-1	- - - - - - 0 -
CMPA (opr)	Compare A to Memory	A − M	A IMM	81	ii	2	2	3-1	- - - - ↕ ↕ ↕ ↕
			A DIR	91	dd	2	3	4-1	
			A EXT	B1	hh ll	3	4	5-2	
			A IND,X	A1	ff	2	4	6-2	
			A IND,Y	18 A1	ff	3	5	7-2	

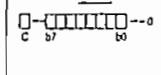
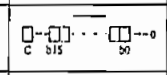
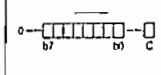
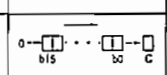
*Cycle-by-cycle number provides a reference to Tables 10-2 through 10-8 which detail cycle-by-cycle operation.
Example: Table 10-1 Cycle-by-Cycle column reference number 2-4 equals Table 10-2 line item 2-4.

Table 10-1. MC68HC11A8 Instructions, Addressing Modes, and Execution Times (Sheet 3 of 7)

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		Bytes	Cycle	Cycle by Cycle*	Condition Codes
				Opcode	Operand(s)				S X H I N Z V C
CMPB (opr)	Compare B to Memory	B – M	B IMM	C1	ii	2	2	3-1	- - - - ↕ ↕ ↕ ↕
			B DIR	D1	dd	2	3	4-1	
			B EXT	F1	hh ll	3	4	5-2	
			B IND,X	E1	ff	2	4	6-2	
			B IND,Y	18 E1	ff	3	5	7-2	
COM (opr)	1's Complement Memory Byte	SFF – M → M	EXT	73	hh ll	3	6	5-8	- - - - ↕ ↕ 0 1
			IND,X	63	ff	2	6	6-3	
			IND,Y	18 63	ff	3	7	7-3	
COMA	1's Complement A	SFF – A → A	A INH	43		1	2	2-1	- - - - ↕ ↕ 0 1
COMB	1's Complement B	SFF – B → B	B INH	53		1	2	2-1	- - - - ↕ ↕ 0 1
CPD (opr)	Compare D to Memory 16-Bit	D – M:M + 1	IMM	1A 83	jj kk	4	5	3-5	- - - - ↕ ↕ ↕ ↕
			DIR	1A 93	dd	3	6	4-9	
			EXT	1A B3	hh ll	4	7	5-11	
			IND,X	1A A3	ff	3	7	6-11	
			IND,Y	CD A3	ff	3	7	7-8	
CPX (opr)	Compare X to Memory 16-Bit	IX – M:M + 1	IMM	8C	jj kk	3	4	3-3	- - - - ↕ ↕ ↕ ↕
			DIR	9C	dd	2	5	4-7	
			EXT	BC	hh ll	3	6	5-10	
			IND,X	AC	ff	2	6	6-10	
			IND,Y	CD AC	ff	3	7	7-8	
CPY (opr)	Compare Y to Memory 16-Bit	IY – M:M + 1	IMM	18 8C	jj kk	4	5	3-5	- - - - ↕ ↕ ↕ ↕
			DIR	18 9C	dd	3	6	4-9	
			EXT	18 BC	hh ll	4	7	5-11	
			IND,X	1A AC	ff	3	7	6-11	
			IND,Y	18 AC	ff	3	7	7-8	
DAA	Decimal Adjust A	Adjust Sum to BCD	INH	19		1	2	2-1	- - - - ↕ ↕ ↕ ↕
DEC (opr)	Decrement Memory Byte	M – 1 → M	EXT	7A	hh ll	3	6	5-8	- - - - ↕ ↕ ↕ -
			IND,X	6A	ff	2	6	6-3	
			IND,Y	18 6A	ff	3	7	7-3	
DECA	Decrement Accumulator A	A – 1 → A	A INH	4A		1	2	2-1	- - - - ↕ ↕ ↕ -
DECB	Decrement Accumulator B	B – 1 → B	B INH	5A		1	2	2-1	- - - - ↕ ↕ ↕ -
DES	Decrement Stack Pointer	SP – 1 → SP	INH	34		1	3	2-3	- - - - - - - -
DEX	Decrement Index Register X	IX – 1 → IX	INH	09		1	3	2-2	- - - - - ↕ - -
DEY	Decrement Index Register Y	IY – 1 → IY	INH	18 09		2	4	2-4	- - - - - ↕ - -
EORA (opr)	Exclusive OR A with Memory	A ⊕ M → A	A IMM	88	ii	2	2	3-1	- - - - ↕ ↕ 0 -
			A DIR	98	dd	2	3	4-1	
			A EXT	B8	hh ll	3	4	5-2	
			A IND,X	A8	ff	2	4	6-2	
			A IND,Y	18 A8	ff	3	5	7-2	
EORB (opr)	Exclusive OR B with Memory	B ⊕ M → B	B IMM	C8	ii	2	2	3-1	- - - - ↕ ↕ 0 -
			B DIR	D8	dd	2	3	4-1	
			B EXT	F8	hh ll	3	4	5-2	
			B IND,X	E8	ff	2	4	6-2	
			B IND,Y	18 E8	ff	3	5	7-2	
FDIV	Fractional Divide 16 by 16	D/IX → IX; r → D	INH	03		1	41	2-17	- - - - - ↕ ↕ ↕
IDIV	Integer Divide 16 by 16	D/IX → IX; r → D	INH	02		1	41	2-17	- - - - - ↕ 0 ↕
INC (opr)	Increment Memory Byte	M + 1 → M	EXT	7C	hh ll	3	6	5-8	- - - - ↕ ↕ ↕ -
			IND,X	6C	ff	2	6	6-3	
			IND,Y	18 6C	ff	3	7	7-3	
INCA	Increment Accumulator A	A + 1 → A	A INH	4C		1	2	2-1	- - - - ↕ ↕ ↕ -
INCB	Increment Accumulator B	B + 1 → B	B INH	5C		1	2	2-1	- - - - ↕ ↕ ↕ -
INS	Increment Stack Pointer	SP + 1 → SP	INH	31		1	3	2-3	- - - - - - - -

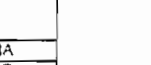
*Cycle-by-cycle number provides a reference to Tables 10-2 through 10-8 which detail cycle-by-cycle operation.
Example: Table 10-1 Cycle-by-Cycle column reference number 2-4 equals Table 10-2 line item 2-4.

Table 10-1. MC68HC11A8 Instructions, Addressing Modes, and Execution Times (Sheet 4 of 7)

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal) Opcode	Operand(s)	Bytes	Cycle	Cycle by Cycle*	Condition Codes S X H I N Z V C
INX	Increment Index Register X	IX + 1 → IX	INH	08		1	3	2-2	- - - - - ↕ - -
INY	Increment Index Register Y	IY + 1 → IY	INH	18 08		2	4	2-4	- - - - - ↕ - -
JMP (opr)	Jump	See Special Ops	EXT	7E	hh ll	3	3	5-1	- - - - - - - -
			IND,X	6E	ff	2	3	6-1	
			IND,Y	18 6E	ff	3	4	7-1	
JSR (opr)	Jump to Subroutine	See Special Ops	DIR	9D	dd	2	5	4-8	- - - - - - - -
			EXT	BD	hh ll	3	6	5-12	
			IND,X	AD	ff	2	6	6-12	
			IND,Y	18 AD	ff	3	7	7-9	
LDAA (opr)	Load Accumulator A	M → A	A IMM	86	ii	2	2	3-1	- - - - ↕ ↕ 0 -
			A DIR	96	dd	2	3	4-1	
			A EXT	B6	hh ll	3	4	5-2	
			A IND,X	A6	ff	2	4	6-2	
			A IND,Y	18 A6	ff	3	5	7-2	
LDAB (opr)	Load Accumulator B	M → B	B IMM	C6	ii	2	2	3-1	- - - - ↕ ↕ 0 -
			B DIR	D6	dd	2	3	4-1	
			B EXT	F6	hh ll	3	4	5-2	
			B IND,X	E6	ff	2	4	6-2	
			B IND,Y	18 E6	ff	3	5	7-2	
LDD (opr)	Load Double Accumulator D	M → A,M + 1 → B	IMM	CC	jj kk	3	3	3-2	- - - - ↕ ↕ 0 -
			DIR	DC	dd	2	4	4-3	
			EXT	FC	hh ll	3	5	5-4	
			IND,X	EC	ff	2	5	6-6	
			IND,Y	18 EC	ff	3	6	7-6	
LDS (opr)	Load Stack Pointer	M:M + 1 → SP	IMM	8E	jj kk	3	3	3-2	- - - - ↕ ↕ 0 -
			DIR	9E	dd	2	4	4-3	
			EXT	BE	hh ll	3	5	5-4	
			IND,X	AE	ff	2	5	6-6	
			IND,Y	18 AE	ff	3	6	7-6	
LDX (opr)	Load Index Register X	M:M + 1 → IX	IMM	CE	jj kk	3	3	3-2	- - - - ↕ ↕ 0 -
			DIR	DE	dd	2	4	4-3	
			EXT	FE	hh ll	3	5	5-4	
			IND,X	EE	ff	2	5	6-6	
			IND,Y	CD EE	ff	3	6	7-6	
LDY (opr)	Load Index Register Y	M:M + 1 → IY	IMM	18 CE	jj kk	4	4	3-4	- - - - ↕ ↕ 0 -
			DIR	18 DE	dd	3	5	4-5	
			EXT	18 FE	hh ll	4	6	5-6	
			IND,X	1A EE	ff	3	6	6-7	
			IND,Y	18 EE	ff	3	6	7-6	
LSL (opr)	Logical Shift Left		EXT	78	hh ll	3	6	5-8	- - - - ↕ ↕ ↕ ↕
			IND,X	68	ff	2	6	6-3	
			IND,Y	18 68	ff	3	7	3-7	
LSLA		C b7 b0	A INH	48		1	2	2-1	
LSLB			B INH	58		1	2	2-1	
LSLD	Logical Shift Left Double		INH	05		1	3	2-2	- - - - ↕ ↕ ↕ ↕
		C b15 b0							
LSR (opr)	Logical Shift Right		EXT	74	hh ll	3	6	5-8	- - - - 0 ↕ ↕ ↕
			IND,X	64	ff	2	6	6-3	
			IND,Y	18 64	ff	3	7	7-3	
LSRA		0 b7 b0 C	A INH	44		1	2	2-1	
LSRB			B INH	54		1	2	2-1	
LSRD	Logical Shift Right Double		INH	04		1	3	2-2	- - - - 0 ↕ ↕ ↕
		0 b15 b0 C							
MUL	Multiply 8 by 8	AxB → D	INH	3D		1	10	2-13	- - - - - - - ↕

*Cycle-by-cycle number provides a reference to Tables 10-2 through 10-8 which detail cycle-by-cycle operation.
Example: Table 10-1 Cycle-by-Cycle column reference number 2-4 equals Table 10-2 line item 2-4.

Table 10-1. MC68HC11A8 Instructions, Addressing Modes, and Execution Times (Sheet 5 of 7)

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal) Opcode	Operand(s)	Bytes	Cycle	Cycle by Cycle*	Condition Codes S X H I N Z V C
NEG (opr)	2's Complement Memory Byte	0 − M → M	EXT	70	hh ll	3	6	5-8	- - - - ↕ ↕ ↕ ↕
			IND,X	60	ff	2	6	6-3	
			IND,Y	18 60	ff	3	7	7-3	
NEGA	2's Complement A	0 − A → A	A INH	40		1	2	2-1	- - - - ↕ ↕ ↕ ↕
NEGB	2's Complement B	0 − B → B	B INH	50		1	2	2-1	- - - - ↕ ↕ ↕ ↕
NOP	No Operation	No Operation	INH	01		1	2	2-1	- - - - - - - -
ORAA (opr)	OR Accumulator A (Inclusive)	A + M → A	A IMM	8A	ll	2	2	3-1	- - - - ↕ ↕ 0 -
			A DIR	9A	dd	2	3	4-1	
			A EXT	BA	hh ll	3	4	5-2	
			A IND,X	AA	ff	2	4	6-2	
			A IND,Y	18 AA	ff	3	5	7-2	
ORAB (opr)	OR Accumulator B (Inclusive)	B + M → B	B IMM	CA	ii	2	2	3-1	- - - - ↕ ↕ 0 -
			B DIR	DA	dd	2	3	4-1	
			B EXT	FA	hh ll	3	4	5-2	
			B IND,X	EA	ff	2	4	6-2	
			B IND,Y	18 EA	ff	3	5	7-2	
PSHA	Push A onto Stack	A → Stk, SP = SP−1	A INH	36		1	3	2-6	- - - - - - - -
PSHB	Push B onto Stack	B → Stk, SP = SP−1	B INH	37		1	3	2-6	- - - - - - - -
PSHX	Push X onto Stack (Lo First)	IX → Stk, SP = SP−2	INH	3C		1	4	2-7	- - - - - - - -
PSHY	Push Y onto Stack (Lo First)	IY → Stk, SP = SP−2	INH	18 3C		2	5	2-8	- - - - - - - -
PULA	Pull A from Stack	SP = SP + 1, A←Stk	A INH	32		1	4	2-9	- - - - - - - -
PULB	Pull B from Stack	SP = SP + 1, B←Stk	B INH	33		1	4	2-9	- - - - - - - -
PULX	Pull X from Stack (Hi First)	SP = SP + 2, IX←Stk	INH	38		1	5	2-10	- - - - - - - -
PULY	Pull Y from Stack (Hi First)	SP = SP + 2, IY←Stk	INH	18 38		2	6	2-11	- - - - - - - -
ROL (opr)	Rotate Left		EXT	79	hh ll	3	6	5-8	- - - - ↕ ↕ ↕ ↕
			IND,X	69	ff	2	6	6-3	
			IND,Y	18 69	ff	3	7	7-3	
ROLA			A INH	49		1	2	2-1	
ROLB			B INH	59		1	2	2-1	
ROR (opr)	Rotate Right		EXT	76	hh ll	3	6	5-8	- - - - ↕ ↕ ↕ ↕
			IND,X	66	ff	2	6	6-3	
			IND,Y	18 66	ff	3	7	7-3	
RORA			A INH	46		1	2	2-1	
RORB			B INH	56		1	2	2-1	
RTI	Return from Interrupt	See Special Ops	INH	3B		1	12	2-14	↕ ↕ ↕ ↕ ↕ ↕ ↕ ↕ ↕
RTS	Return from Subroutine	See Special Ops	INH	39		1	5	2-12	- - - - - - - -
SBA	Subtract B from A	A − B → A	INH	10		1	2	2-1	- - - - ↕ ↕ ↕ ↕
SBCA (opr)	Subtract with Carry from A	A − M − C → A	A IMM	82	ll	2	2	3-1	- - - - ↕ ↕ ↕ ↕
			A DIR	92	dd	2	3	4-1	
			A EXT	B2	hh ll	3	4	5-2	
			A IND,X	A2	ff	2	4	6-2	
			A IND,Y	18 A2	ff	3	5	7-2	
SBCB (opr)	Subtract with Carry from B	B − M − C → B	B IMM	C2	ii	2	2	3-1	- - - - ↕ ↕ ↕ ↕
			B DIR	D2	dd	2	3	4-1	
			B EXT	F2	hh ll	3	4	5-2	
			B IND,X	E2	ff	2	4	6-2	
			B IND,Y	18 E2	ff	3	5	7-2	
SEC	Set Carry	1 → C	INH	0D		1	2	2-1	- - - - - - - 1
SEI	Set Interrupt Mask	1 → I	INH	0F		1	2	2-1	- - - 1 - - - -
SEV	Set Overflow Flag	1 → V	INH	0B		1	2	2-1	- - - - - - 1 -

*Cycle-by-cycle number provides a reference to Tables 10-2 through 10-8 which detail cycle-by-cycle operation.
Example: Table 10-1 Cycle-by-Cycle column reference number 2-4 equals Table 10-2 line item 2-4.

Table 10-1. MC68HC11A8 Instructions, Addressing Modes, and Execution Times (Sheet 6 of 7)

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal) Opcode	Operand(s)	Bytes	Cycle	Cycle by Cycle*	Condition Codes S X H I N Z V C
STAA (opr)	Store Accumulator A	A → M	A DIR	97	dd	2	3	4-2	- - - - ↕ ↕ 0 -
			A EXT	B7	hh ll	3	4	5-3	
			A IND,X	A7	ff	2	4	6-5	
			A IND,Y	18 A7	ff	3	5	7-5	
STAB (opr)	Store Accumulator B	B → M	B DIR	D7	dd	2	3	4-2	- - - - ↕ ↕ 0 -
			B EXT	F7	hh ll	3	4	5-3	
			B IND,X	E7	ff	2	4	6-5	
			B IND,Y	18 E7	ff	3	5	7-5	
STD (opr)	Store Accumulator D	A → M, B → M + 1	DIR	DD	dd	2	4	4-4	- - - - ↕ ↕ 0 -
			EXT	FD	hh ll	3	5	5-5	
			IND,X	ED	ff	2	5	6-8	
			IND,Y	18 ED	ff	3	6	7-7	
STOP	Stop Internal Clocks		INH	CF		1	2	2-1	- - - - - - - -
STS (opr)	Store Stack Pointer	SP → M:M + 1	DIR	9F	dd	2	4	4-4	- - - - ↕ ↕ 0 -
			EXT	BF	hh ll	3	5	5-5	
			IND,X	AF	ff	2	5	6-8	
			IND,Y	18 AF	ff	3	6	7-7	
STX (opr)	Store Index Register X	IX → M:M + 1	DIR	DF	dd	2	4	4-4	- - - - ↕ ↕ 0 -
			EXT	FF	hh ll	3	5	5-5	
			IND,X	EF	ff	2	5	6-8	
			IND,Y	CD EF	ff	3	6	7-7	
STY (opr)	Store Index Register Y	IY → M:M + 1	DIR	18 DF	dd	3	5	4-6	- - - - ↕ ↕ 0 -
			EXT	18 FF	hh ll	4	6	5-7	
			IND,X	1A EF	ff	3	6	6-9	
			IND,Y	18 EF	ff	3	6	7-7	
SUBA (opr)	Subtract Memory from A	A − M → A	A IMM	80	ii	2	2	3-1	- - - - ↕ ↕ ↕ ↕
			A DIR	90	dd	2	3	4-1	
			A EXT	B0	hh ll	3	4	5-2	
			A IND,X	A0	ff	2	4	6-2	
			A IND,Y	18 A0	ff	3	5	7-2	
SUBB (opr)	Subtract Memory from B	B − M → B	B IMM	C0	ii	2	2	3-1	- - - - ↕ ↕ ↕ ↕
			B DIR	D0	dd	2	3	4-1	
			B EXT	F0	hh ll	3	4	5-2	
			B IND,X	E0	ff	2	4	6-2	
			B IND,Y	18 E0	ff	3	5	7-2	
SUBD (opr)	Subtract Memory from D	D − M:M + 1 → D	IMM	83	jj kk	3	4	3-3	- - - - ↕ ↕ ↕ ↕
			DIR	93	dd	2	5	4-7	
			EXT	B3	hh ll	3	6	5-10	
			IND,X	A3	ff	2	6	6-10	
			IND,Y	18 A3	ff	3	7	7-8	
SWI	Software Interrupt	See Special Ops	INH	3F		1	14	2-15	- - - 1 - - - -
TAB	Transfer A to B	A → B	INH	16		1	2	2-1	- - - - ↕ ↕ 0 -
TAP	Transfer A to CC Register	A → CCR	INH	06		1	2	2-1	↕ ↕ ↕ ↕ ↕ ↕ ↕ ↕
TBA	Transfer B to A	B → A	INH	17		1	2	2-1	- - - - ↕ ↕ 0 -
TEST	TEST (Only in Test Modes)	Address Bus Counts	INH	00		1	**	2-20	- - - - - - - -
TPA	Transfer CC Register to A	CCR → A	INH	07		1	2	2-1	- - - - - - - -
TST (opr)	Test for Zero or Minus	M − 0	EXT	7D	hh ll	3	6	5-9	- - - - ↕ ↕ 0 0
			IND,X	6D	ff	2	6	6-4	
			IND,Y	18 6D	ff	3	7	7-4	
TSTA		A − 0	A INH	4D		1	2	2-1	- - - - ↕ ↕ 0 0
TSTB		B − 0	B INH	5D		1	2	2-1	- - - - ↕ ↕ 0 0
TSX	Transfer Stack Pointer to X	SP + 1 → IX	INH	30		1	3	2-3	- - - - - - - -
TSY	Transfer Stack Pointer to Y	SP + 1 → IY	INH	18 30		2	4	2-5	- - - - - - - -

*Cycle-by-cycle number provides a reference to Tables 10-2 through 10-8 which detail cycle-by-cycle operation.
Example: Table 10-1 Cycle-by-Cycle column reference number 2-4 equals Table 10-2 line item 2-4.

Table 10-1. MC68HC11A8 Instructions, Addressing Modes, and Execution Times (Sheet 7 of 7)

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		Bytes	Cycle	Cycle by Cycle*	Condition Codes
				Opcode	Operand(s)				S X H I N Z V C
TXS	Transfer X to Stack Pointer	IX – 1 → SP	INH	35		1	3	2-2	- - - - - - - -
TYS	Transfer Y to Stack Pointer	IY – 1 → SP	INH	18 35		2	4	2-4	- - - - - - - -
WAI	Wait for Interrupt	Stack Regs & WAIT	INH	3E		1	***	2-16	- - - - - - - -
XGDX	Exchange D with X	IX → D, D → IX	INH	8F		1	3	2-2	- - - - - - - -
XGDY	Exchange D with Y	IY → D, D → IY	INH	18 8F		2	4	2-4	- - - - - - - -

*Cycle-by-cycle number provides a reference to Tables 10-2 through 10-8 which detail cycle-by-cycle operation.
Example: Table 10-1 Cycle-by-Cycle column reference number 2-4 equals Table 10-2 line item 2-4.

**Infinity or Until Reset Occurs

***12 Cycles are used beginning with the opcode fetch. A wait state is entered which remains in effect for an integer number of MPU E-clock cycles (n) until an interrupt is recognized. Finally, two additional cycles are used to fetch the appropriate interrupt vector (14 + n total).

dd = 8-Bit Direct Address ($0000 –$00FF) (High Byte Assumed to be $00)
ff = 8-Bit Positive Offset $00 (0) to $FF (255) (Is Added to Index)
hh = High Order Byte of 16-Bit Extended Address
ii = One Byte of Immediate Data
jj = High Order Byte of 16-Bit Immediate Data
kk = Low Order Byte of 16-Bit Immediate Data
ll = Low Order Byte of 16-Bit Extended Address
mm = 8-Bit Bit Mask (Set Bits to be Affected)
rr = Signed Relative Offset $80 (– 128) to $7F (+ 127)
(Offset Relative to the Address Following the Machine Code Offset Byte)