2021

Proyecto Ensamblador

ARQUITECTURA DE COMPUTADORES

PABLO FERNÁNDEZ DE GRACIA 200068
ÁLVARO CABO CIUDAD 200172

Índice

Diagrama de flujo o pseudocódigo y comentario de los algoritmos utilizados

- INIT
  La idea de esta subrutina es crear una especie de constructor que permita inicializar todo lo necesario para realizar la transmisión, tanto los punteros de lectura, escritura, el vector de interrupciones y como los parámetros necesarios para esta.

  Al ser una subrutina de inicio, podríamos plantearla como un constructor sin parámetros, ya que todos están predefinidos

```
INIT(){
        //Linea A (Operamos con números enteros que son instancias de binarios)
        CRA= 32;
        MR1A=3;
        MR2A=0;
        //Repetimos para línea B
        // Control de velocidad DUART
        // Inicio recepcion
                CRA= CRA+5;
                CRB= CRB+5;
        //Creacion vector director
        Private IVR = 128;
        RTI -> 100 //Guardamos la direccion #0´100 en $RTI
        IMR = 66
        Copia-IMR =IMR //Guardamos el puntero a IMR en la copia
INI-BUFS();
}
```

- SCAN(Dir-Buffer, Descriptor, Tamaño){

```
        //Salvamos a pila
        If(tamaño==0) return 0;
        Int contador=0;
        If(Descriptor ==0) Print_AA(contador);
        Else if(Descriptor==1)    Print_BB(contador);
        Else return –1;
    }
    SCAN_AA{
    while(Leecar(0)!=-1 or Contador==Tamanho){
        Contador++;
        St(Leecar(0)) -> buffer +contador;
        }
    Return contador;
    }
    //SCAN_BB es equivalente a SCAN_AA salvo por el parametro de entrada a leecar (1)
```

- PRINT{

```
    //Salvamos a pila
        If(tamaño==0) return 0;
        Int contador=0; int res:
        If(Descriptor ==0) res=Print_AA();
        Else if(Descriptor==1)    res=Print_BB();
        Else res= –1;
        Return res;
}
```

```
PRINT_AA{
while(Escaccar(0)==0 or Contador==Tamanho){
        Contador++;
        St(esccar(3)) -> buffer +contador;
        }
//Desactivamos la interrupciones por esa linea
        Return contador;
}
```

//PRINT_BB es equivalente a PRINT_AA salvo por el parametro de entrada a escacar(4);

- RTI(){
  $puerta = ISR AND COPY_IMR

  Switch(puerta){

  Case 0: Trans_A(); BREAK;

  Case 1: Recep_A(); BREAK;

  Case 4: Trans_B(); BREAK;

  Case 5: Recep_B(); BREAK;

  Default:

  //Recuperamos D0-D2 de la pila

  //Recuperamos PC

  }

  //Reservamos

- TRANS_A {
  If (Leecar(2)==-1) {
  D2=COPIA_IMR; //desactivar interrupciones
  IMR[0]=0; //de transferencia por A
  }
  Else {
  D0=Leecar(2);
  TBA = D0;
  }
  Return D0;

}

- TRANS_B {
  If (Leecar(3)==-1) {
  D2=COPIA_IMR; //desactivar interrupciones
  IMR[4]=0; //de transferencia por B
  }
  Else {
  D0=Leecar(3);
  TBB = D0;
  }
  Return D0;

}

- RECEP_A {
  D0 = 0;
  D1 = RBA;
  If (Esccar(D0)==-1) {
  Llamada a RTI;
  }
  Else {
  Llamada a RTI_FIN;
  }

  }

- RECEP_B {

  D0 = 1;

  D1 = RBB;

  If (Esccar(D0)==-1) {

  Llamada a RTI;

  }

  Else {

  Llamada a RTI_FIN;

  }

  }

Listado comentado de las subrutinas en ensamblador

INIT ()

```
INIT    *Iniciamos A
        MOVE.B          #%00010000,CRA      * Reinicia el puntero de control CRA para
acceder a MR1A
        MOVE.B          #%00000011,MR1A     * 8 bits por caracter en A. Solicita una
interrupcion cada caracter
        MOVE.B          #%00000000,MR2A     * Eco desactivado en A
        *Iniciamos B
        MOVE.B          #%00010000,CRB      * Reinicia el puntero de control CRB para
acceder a MR1B
        MOVE.B          #%00000011,MR1B     * 8 bits por caracter en B. Solicita una
interrupcion cada caracter
        MOVE.B          #%00000000,MR2B     * Eco desactivado en B

        *Control de la DUART
        MOVE.B          #%00000000,ACR      * Seleccionamos velocidad conjunto 1 =
38400 bps
        MOVE.B          #%11001100,CSRA     * Velocidad = 38400 bps (tranmision y
recepcion)
        MOVE.B          #%11001100,CSRB     * Velocidad = 38400 bps (tranmision y
recepcion)
        *Inicio de recepción
        MOVE.B          #%00010101,CRA      * Activamos transmision y recepcion para
A
        MOVE.B          #%00010101,CRB      * Activamos transmision y recepcion para
B
        *Control de interrupciones
        MOVE.B          #%01000000,IVR      * Estableces vector de Interrupcion 40
Hexadecimal
        MOVE.L          #RTI,$100               * Colocamos la dirección de PC de la
RTI en el
                                                * inicio de los vectores de
interrupción
        MOVE.B          #%00100010,IMR          * Solo permitimos las recepciones
        MOVE.B          #%00100010,COPIA_IMR    * Actualiza copia de IMR

        BSR INI_BUFS                        * INICIALIZA BUFFERS

        RTS                                 * retorno.
```

PRINT (Buffer, Descriptor, Tamaño)

```
PRINT       LINK     A6,#-4          * Marco de pila para guardar la posición de A7

            MOVEM.L A0-A5/D1-D7,-(A7) * metemos por pila los parámetros a utilizar
            MOVE.L   8(A6),A2         * A2 = buffer
            MOVE.L   #0,D3
            MOVE.W   12(A6),D3        * D3 = descriptor
            MOVE.L   #0,D2
            MOVE.W   14(A6),D2        * D2 = tamaño
            CMP.W    #0,D2            * si tamaño == 0 saltamos PrintZ
            BEQ      PRINTZ
            MOVE.L   #0,D5            * contador
            CMP.W    #0,D3            * comprobamos descriptor
            BEQ      PRINT_AA         * si descriptor == 0 saltamos a PRINT_A
            CMP.W    #1,D3
            BEQ      PRINT_BB         * si descriptor == 1 saltamos a PRINT_B.
            MOVE.L   #-1,D0           * en otro caso, D0 = -1
            BRA      FIN_PRINT
PRINTZ      MOVE.L   #0,D0


FIN_PRINT   MOVEM.L (A7)+,A0-A5/D1-D7 * sacamos parámetros utilizados de la pila
            UNLK     A6
            RTS
PRINT_AA:   MOVE.B (A2)+,D1
            MOVE.L  #2,D0            * parámetros de entrada de ESCCAR
            BSR ESCCAR                  * escribe un dato y lo pone en D0
            CMP.L #0,D0                 * se comprueba que no haya errores
            BNE FA_PRINT                * si son diferentes, hay fallo

            ADD.L #1,D5                 * contador++
            CMP.W D5,D2                 * COMPARO LA PARTE BAJA
            BEQ FA_PRINT                * si he leído todos, salto a fin
            BRA PRINT_AA                * si no, seguimos escribiendo.

****GESTIÓN DE ERRORES****
FA_PRINT    MOVE.L D5,D0                * D0 = caracteres escritos
            MOVE.B COPIA_IMR,D3         * D3 = COPIA_IMR
            BSET #0,D3
            MOVE.B D3,COPIA_IMR
            MOVE.B D3,IMR               * actualizo IMR
            BRA FIN_PRINT
```

```
PRINT_BB:   MOVE.B (A2)+,D1
            MOVE.L #3,D0                * parametros de entrada de ESCCAR
            BSR ESCCAR
            CMP.L #0,D0                 * comprobamos todo ok
            BNE FB_PRINT                * si no saltamos a FB_PRINT

            ADD.L #1,D5                 * contador++
            CMP.W D5,D2                 * COMPARO LA PARTE BAJA
            BEQ FB_PRINT                * si he transmitido todo, salto a fin
            BRA PRINT_BB                * si no, seguimos escribiendo

****GESTIÓN DE ERRORES****
FB_PRINT    MOVE.L D5,D0                * D0 = caracteres escritos
            MOVE.B COPIA_IMR,D3         * D3 = COPIA_IMR
            BSET #4,D3
            MOVE.B D3,COPIA_IMR
            MOVE.B D3,IMR               * actualizo IMR
            BRA FIN_PRINT
```

SCAN (Buffer, Descriptor, Tamaño)

```
SCAN        LINK     A6,#-4            * Marco de pila para guardar la posición de A7
                                       * y meter los parámetros a la subrutina
            MOVEM.L A0-A5/D1-D7,-(A7) * metemos por pila los parámetros a utilizar
            MOVE.L   8(A6),A5          * A5 = buffer
            MOVE.L   #0,D4
            MOVE.W   12(A6),D4         * D4 = descriptor
            MOVE.L   #0,D3
            MOVE.W   14(A6),D3         * D3 = tamaño.
            CMP.W    #0,D3             * si tamaño == 0 saltamos SCANZ
            BEQ      SCANZ
            MOVE.L   #0,D6             * contador
            CMP.W    #0,D4
            BEQ      SC_A              * si descriptor == 0 saltamos a 0.
            CMP.W    #1,D4
            BEQ      SC_B              * si descriptor == 1 saltamos a 1.
            MOVE.L   #-1,D0            * en otro caso, D0 = -1
            BRA      F_SCAN
SCANZ       MOVE.L   #0,D0
F_SCAN
            MOVEM.L (A7)+,A0-A5/D1-D7 * sacamos parámetros utilizados de la pila
            UNLK     A6
            RTS


SC_A:       MOVE.L   #0,D0           * parametro de entrada de LEECAR
            BSR      LEECAR
            CMP.B #-1,D0             * Está vacío, salto a F_SCA para que
            BEQ F_SCA                * D0 <- Nº caracteres leidos

            ADD.L #1,D6              * contador++
            MOVE.B D0,(A5)+          * Meto el caracter en el buffer
            CMP.W D6,D3
            BNE SC_A                 * si datos leídos != total => salto a SC_A
F_SCA       MOVE.L   D6,D0           * devuelvo en D0 caracteres leídos
            BRA F_SCAN
```

```
SC_B:       MOVE.L   #1,D0          * parametro de entrada de LEECAR
            BSR      LEECAR

            CMP.B #$ffffffff,D0     * Está vacío, salto a F_SCB para que
            BEQ      F_SCB          * D0 <- Nº caracteres leidos

            ADD.L #1,D6             * contador++
            MOVE.B D0,(A5)+
            CMP.W D6,D3
            BNE SC_B                * si datos leídos != total => salto a SC_B

            MOVE.L   D6,D0
            BRA F_SCAN              * si no, fin

F_SCB       MOVE.L   D6,D0          * devuelvo en D0 caracteres leídos
            BRA F_SCAN
```

RTI

```
RTI         * En vez de meter directamente a pila,
            * creamos un marco de pila para guardar los registros (Más cómodo
            * debido a la gran cantidad de procesos que dependen de la RTI).

            MOVEM.L  D0-D2,-(A7)    *Solo necesitamos 2 registros para hacer toda la
operación

RTI_LOOP:
            MOVE.B   ISR,D1             * D2 <- ISR
            AND.B    COPIA_IMR,D1       * D2 <- Periferico que interrumpe
            *Interrupciones de Recepción (FIFO no vacía)
            BTST     #0,D1
            BNE      TRANS_A            * si el bit 0 TxRDYA = 1 => saltamos a
transmision por linea A
            BTST     #1,D1
            BNE      RECEP_A            * si el bit 1 RxRDYA= 1 => recepcion por
linea A
            BTST     #4,D1
            BNE      TRANS_B            * si el bit 4 TxRDYB = 1 =>  saltamos a
transmision por linea A
            BTST     #5,D1
            BNE      RECEP_B            * si el bit 5 RxRDYB = 1 =>  a recepcion por
linea B

FIN_RTI
            MOVEM.L   (A7)+,D0-D2
            RTE                         * PC <- Old PC
```

TRANS_A

```
TRANS_A     MOVE.L   #2,D0                  * LEECAR (2 => Trans_A)
            BSR LEECAR

            CMP.B #-1,D0                    * si D0 = -1 -> Fin transmisión
            BEQ TRANS_AA

            *Si no da error, lo guardamos en el buffer y terminamos la interrupción
            MOVE.B D0,TBA
            BRA FIN_RTI
TRANS_AA
            MOVE.B COPIA_IMR,D2             * Desactivamos las interrupciones
            BCLR #0,D2                      * De transferencia por A
            MOVE.B D2,COPIA_IMR             * Update Copia IMR
            MOVE.B D2,IMR                   * Update IMR
            BRA FIN_RTI
```

TRANS_B

```
TRANS_B     MOVE.L   #3,D0                  * parametro de entrada de LEECAR es
3(buffer B de transmision)
            BSR LEECAR                      * lee 1 dato y lo pone en D0

            CMP.B #-1,D0                    * si D0 == -1 deshabilitar interrupciones
de transmision linea B
            BEQ TRANS_BB

            *Si no da error, lo guardamos en el buffer
            MOVE.B D0,TBB                   * Caracter leido = @param: TBB.
            BRA FIN_RTI
TRANS_BB
            MOVE.B COPIA_IMR,D2
            BCLR #4,D2                      * IMR[4]=0, => desactivamos interrupcion.
            MOVE.B D2,COPIA_IMR             * Update Copia IMR
            MOVE.B D2,IMR                   * Update IMR
            BRA FIN_RTI
```

RECEP_A

```
RECEP_A     MOVE.L #0,D0                    * ESCCAR (0)* Puerto A
            MOVE.B RBA,D1                   * D1 = Registro Buffer A
            BSR ESCCAR                      * escribe el dato en el buffer
            CMP.L #-1,D0                    * Si D0 == -1, no salimos de la RTI
            BNE RTI_LOOP
            BRA FIN_RTI                     * salto a FIN_RTI.
```

RECEP_B

```
RECEP_B     MOVE.L  #1,D0                   * parametro de entrada de ESCCAR es
1(buffer B de recepcion)
            MOVE.B RBB,D1                   * llevamos el registro de buffer de
recepcion de A a D1
            BSR ESCCAR                      * escribe el dato en el buffer
            CMP.L #-1,D0
            BNE RTI_LOOP
            BRA FIN_RTI                     * salto a FIN_RTI.              * salto a
FIN_RTI.
```

Descripción del juego de ensayo (conjunto de casos de prueba)

Para realizar las pruebas, inicialmente probábamos con cadenas de caracteres arbitrarios para asegurarnos de que se transmitía y se escaneaba de manera correcta. Posteriormente, tras realizar las entregas nos descargábamos el fichero de correcciones e íbamos probando una a una las pruebas que fallaban sustituyendo los datos de una estructura como la siguiente:

```
PPAL
 BUFFER:    DS.B 2100                   * Buffer para lectura y escritura de
caracteres
 PARDIR:    DC.L 0                      * Direccion que se pasa como parametro
 PARTAM:    DC.W 0                      * Tamaño que se pasa como parametro
 CONTC:     DC.W 0                      * Contador de caracteres a imprimir
 DESA:      EQU 0                       * Descriptor linea A
 DESB:      EQU 1                       * Descriptor linea B
 TAMBS:     EQU 100                     * Tamaño de bloque para SCAN
 TAMBP:     EQU 102                     * Tamaño de bloque para PRINT


PRUEBA: DC.B ‘12345678901234567890123456789012345678901234567890’
        DC.B ‘12345678901234567890123456789012345678901234567890’
        DC.B 0x0D

 * Manejadores de excepciones
 INICIO:
            MOVE.L  #$8000,A7
            MOVE.L  #$0,D0
            MOVE.L  #$0,D1
            MOVE.L  #$0,D2
            MOVE.L  #$0,D3
            MOVE.L  #$0,D4
            MOVE.L  #$0,D5
            MOVE.L  #$0,D6
            MOVE.L  #$0,D7
            MOVE.L  #$0,A0
            MOVE.L  #$0,A1
            MOVE.L  #$0,A2
            MOVE.L  #$0,A3
            MOVE.L  #$0,A4
            MOVE.L  #$0,A5
            MOVE.L  #$0,A6
```

```
            MOVE.L  #BUS_ERROR,8        * Bus error handler
            MOVE.L  #ADDRESS_ER,12      * Address error handler
            MOVE.L  #ILLEGAL_IN,16      * Illegal instruction handler
            MOVE.L  #PRIV_VIOLT,32      * Privilege violation handler
            MOVE.L  #ILLEGAL_IN,40      * Illegal instruction handler
            MOVE.L  #ILLEGAL_IN,44      * Illegal instruction handler

            BSR INIT
            MOVE.W #$2000,SR            * Permite interrupciones


 BUCPR:     MOVE.W #TAMBS,PARTAM        * Inicializa parametro de tamaño
            MOVE.L #BUFFER,PARDIR       * Parametro BUFFER = comienzo del buffer
 OTRAE:     MOVE.W #TAMBP,PARTAM        * Tamaño de escritura = Tamaño de bloque
 ESPE:
            BREAK
            MOVE.W PARTAM,-(A7)         * Tamaño de escritura
            MOVE.W #DESB,-(A7)          * Puerto B
            PEA PRUEBA
            JSR PRINT
            ADD.L #8,A7
            BREAK               * Restablece la pila
            ADD.L D0,PARDIR             * Calcula la nueva direccion del buffer
            SUB.W D0,CONTC              * Actualiza el contador de caracteres
            BEQ SALIR                   * Si no quedan caracteres se acaba
            SUB.W D0,PARTAM             * Actualiza el tamano de escritura
            BNE ESPE                    * Si no se ha escrito todo el bloque se
insiste
            CMP.W #TAMBP,CONTC          * Si el no de caracteres que quedan es menor
que el tamano establecido se imprime ese numero
            BHI OTRAE                   * Siguiente bloque
            MOVE.W CONTC,PARTAM
            BRA ESPE                    * Siguiente bloque
 SALIR:     BRA BUCPR
 BUS_ERROR:  BREAK                      * Bus error handler
 *          NOP
 ADDRESS_ER: BREAK                      * Address error handler
 *          NOP
 ILLEGAL_IN: BREAK                      * Illegal instruction handler
 *          NOP
 PRIV_VIOLT: BREAK                      * Privilege violation handler
 *          NOP
```

En esta prueba, estábamos probando a la subrutina PRINT con la prueba 36 por ejemplo, en la cual se escribía bloques de 100 caracteres compuestos por la cadena ‘1234567890’ repetida 10 veces.

Observaciones finales y comentarios personales de la práctica, así como la estimación del tiempo empleado en la misma

Tras la elaboración de este proyecto, tanto mi compañero y yo hemos aprendido el manejo del lenguaje ensamblador de un computador con arquitectura CISC, ya que a diferencia del proyecto de Estructura de Computadores (RISC) en este caso las instrucciones eran algo más complejas, e incluso hemos descubierto instrucciones las cuales desconocíamos por completo.

En la convocatoria ordinaria, no conseguimos superar las pruebas mínimas ya que fallaban 3 pruebas de la línea B, debido a que le pasábamos como parámetro un #4 a ESCCAR queriendo indicar el buffer de transmisión de la línea B cuando debería haber sido un #3. Tras la tutoría con el profesor Santiago Rodríguez, detectamos el fallo y conseguimos pasar todas las pruebas del corrector.

Por otra parte, juntando tanto la convocatoria ordinaria como la extraordinaria, estimamos que hemos tardado un total de 45 horas en la elaboración del proyecto, juntando tanto el tiempo de elaboración de código, elaborar casos de prueba y comprobar los resultados, depuración del código en búsqueda de fallos y en las 2 tutorías que tuvimos con los profesores Juan Zamorano y Santiago Rodríguez. Finalmente, este proyecto ha ocupado un total de 379 líneas de código, de las cuales 296 son destinadas a las subrutinas pedidas, mientras que las 83 restantes son destinadas al programa principal donde comprobamos los casos de prueba.