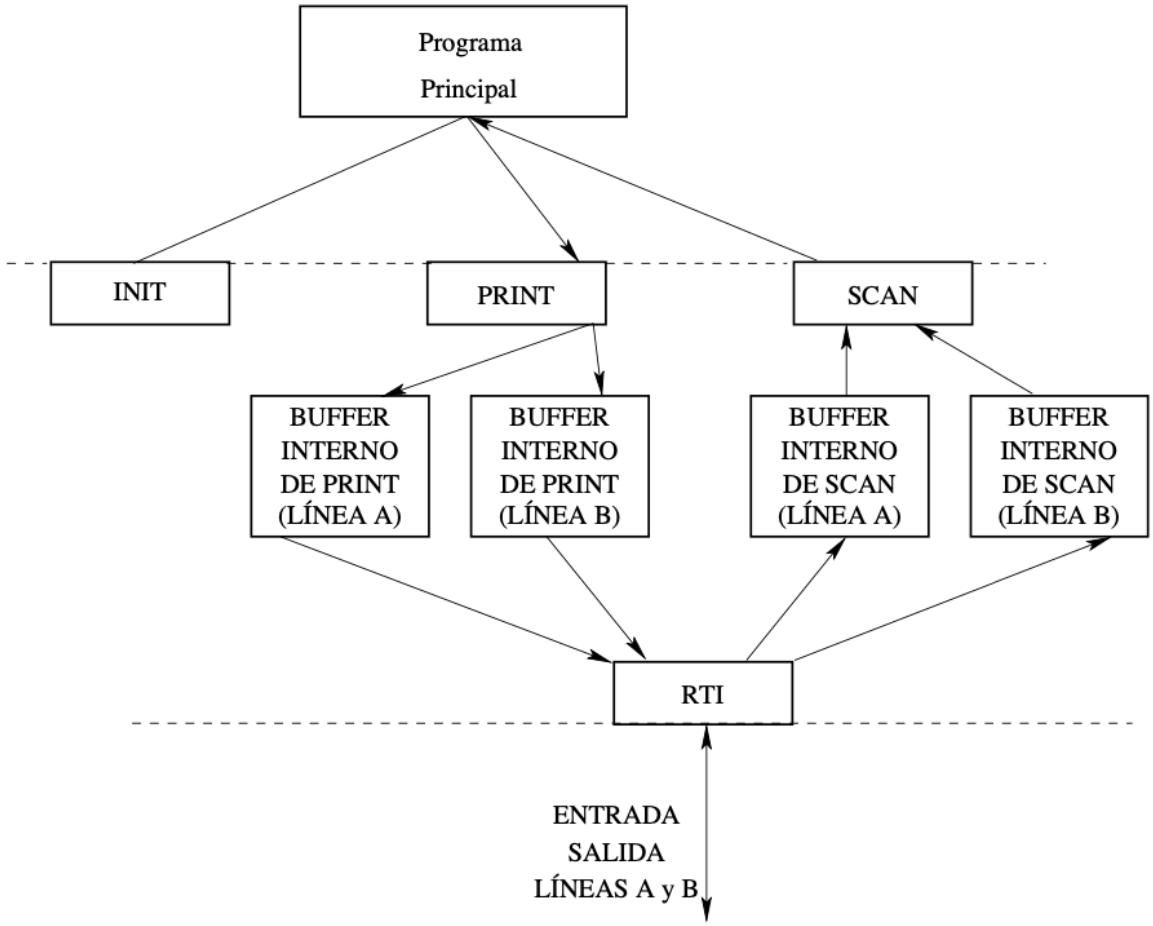
Memoria Proyecto

Arquitectura de computadores

Iñigo Aranguren Redondo
160054
2023

Introducción

La estructura del proyecto es la mostrada a continuación:



Para la llamada a las subrutinas INI_BUFS, LEECAR y ESCCAR se incluirá la biblioteca bib_aux.s al final del fichero de la siguiente forma:

```
INCLUDE    bib_aux.s
```

Además, como el registro de la máscara de interrupción no se puede leer, para conocer el contenido del imr se nos recuerda que se puede mantener una copia en memoria de las escrituras sobre dicho registro, de tal forma, que al principio del programa realizo una declaración de una variable auxiliar para guardar este valor, en la cual guardamos la copia, a esta variable la denomino imrcopia.

INIT

El objetivo de esta subrutina es que las líneas A y B queden preparadas para la recepción y transmisión de caracteres mediante E/S por interrupciones. La rutina INIT no devuelve ningún error por lo tanto no se devuelve ningún valor de retorno. Además, en esta subrutina se hace uso de INI_BUFS.

En esta subrutina se realiza la inicialización de las dos líneas (A y B) según los parámetros dados en el enunciado:

Se reinician punteros

```
MOVE.B  #%00010000,CRA
MOVE.B  #%00010000,CRB
```

Se configuran 8 bits por carácter para ambas lineas

```
MOVE.B  #%00000011,MR1A
MOVE.B  #%00000011,MR1B
```

El eco no debe estar activado en ninguna linea

```
MOVE.B  #%00000000,MR2A
MOVE.B  #%00000000,MR2B
```

Se ajusta la velocidad de recepción y transmisión a la especifica en ambas lineas

```
MOVE.B  #%11001100,CSRA
MOVE.B  #%11001100,CSRB

MOVE.B  #%00000000,ACR
```

Funcionamiento Full duplex, recepción y transmisión activada

```
MOVE.B  #%00000101,CRA
MOVE.B  #%00000101,CRB
```

Inicialización de la mascara de interrupción y de la copia

```
MOVE.B  #%00100010,IMRCOPIA
MOVE.B  IMRCOPIA,IMR
```

El vector de interrupción se establece en 40 hexadecimal = 0x01000000

```
MOVE.B  #%01000000,IVR
```

```
Se actualiza la dirección de la RTI
```

```
MOVE.L  #RTI,$100
```

Llamada a la subrutina INI_BUFS para inicializar los búfferes internos

```
BSR     INI_BUFS
```

SCAN

Esta subrutina realiza la lectura de un bloque de caracteres de la linea que corresponde. El resultado de SCAN se devolverá en D0, donde se almacenará el número de caracteres copiados en buffer, en caso de existiera algún error se devolvería el código 0xFFFFFFFF. En esta subrutina se hace uso de la subrutina LEECAR.

Esta rutina tiene tres parámetros: Buffer, descriptor y tamaño.

El descriptor lo guardo en el registro D1

El tamaño en el registro D2

Se utiliza una variable auxiliar para contar el número de caracteres que se han leído y copiado, esta la guardo en el registro D3

El funcionamiento desarrollado es el siguiente:

Pseudocódigo

- Inicializo los registros que vamos a usar en la subrutina
- Se inicia el bucle de comparación (scancomp)

- Si el tamaño (registro D2) es cero, fin de la subrutina
- Se comprueba el descriptor (registro D1)
- Si es 0 indica que la lectura se realizará en línea A
- Si es 1 indica que la lectura se realizará en línea B
- Si es cualquier otro valor, error, se devuelve el código 0xFFFFFFFF

Scanla

- Se pone D0 a cero para después hacer llamada a LEECAR
- Se llama a bucle (scanbucle) para realizar la lectura

Scanlb

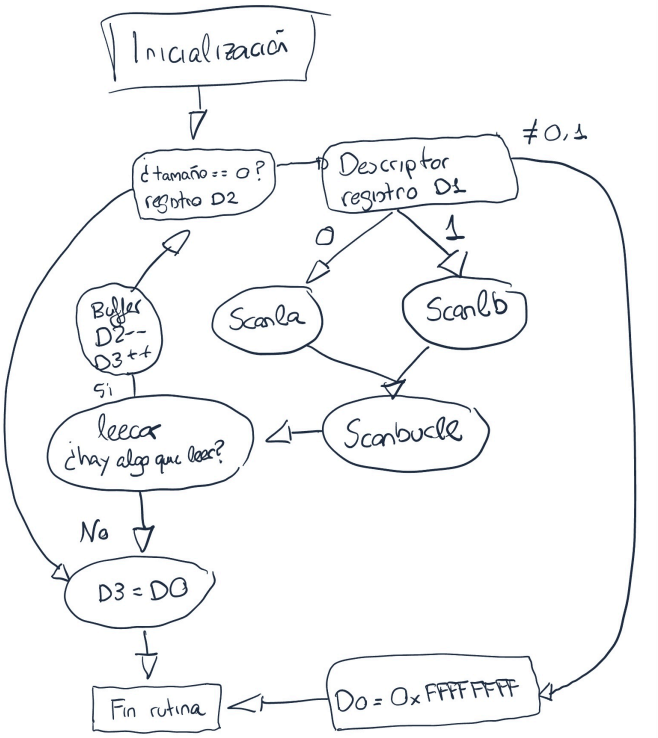
- Se pone D0 a 1 para después hacer llamada a LEECAR
- Se llama a bucle (scanbucle) para realizar la lectura

Scanbucle

- Se llama a LEECAR
- Se comprueba buffer, si no hay nada que leer, finaliza
- Lo devuelto por LEECAR se copia al buffer
- Se decrementa una unidad el tamaño D2
- Se incrementa una unidad el contador D3
- Se vuelve al bucle de comparación (scancomp)

Se devuelve D3 en D0 el resultado

El diagrama de flujo desarrollado aproximado sería el siguiente:



PRINT

Esta subrutina realiza la escritura en el correspondiente buffer. El resultado de PRINT se devuelve en D0, en el cual se devolverá un número positivo indicando el número de caracteres que se han aceptado para escritura. En caso de que en los parámetros pasados exista algún error, se devolverá el código 0xFFFFFFFF. En esta subrutina se hace uso de la subrutina ESCCAR.

Esta rutina tiene tres parámetros: buffer, descriptor y tamaño.

El descriptor lo guardo en el registro D2

El tamaño lo guardo en el D3

Se utiliza una variable auxiliar para contar el número de caracteres que se han aceptado para su escritura, esta la guardo en el registro D4

El funcionamiento desarrollado es el siguiente:

Pseudocódigo

- Inicializo los registros que vamos a usar en la subrutina
- Se comprueba el descriptor (registro D2), si es 0 indica que la escritura se realizará de la linea A
- Se comprueba el descriptor (registro D2), si es 1 indica que la escritura se realizará de la linea B
- Se comprueba si el descriptor es distinto de 0 o 1, si lo es, finalizamos devolviendo el código 0xFFFFFFFF en D0

- Descriptor 0, linea A
- Se comprueba el tamaño registro D3, si es cero, finaliza
- Se llama a ESCCAR
- Se comprueba buffer, si buffer lleno, saltamos a printc
- Se decrementa una unidad el tamaño D3
- Se incrementa una unidad el contador D4
- Se vuelve al bucle comparador

- Descriptor 1, linea B
- Se comprueba el tamaño registro D3, si es cero, finaliza
- Se llama a ESCCAR
- Se comprueba buffer, si buffer lleno, saltamos a printc
- Se decrementa una unidad el tamaño D3
- Se incrementa una unidad el contador D4
- Se vuelve al bucle comparador

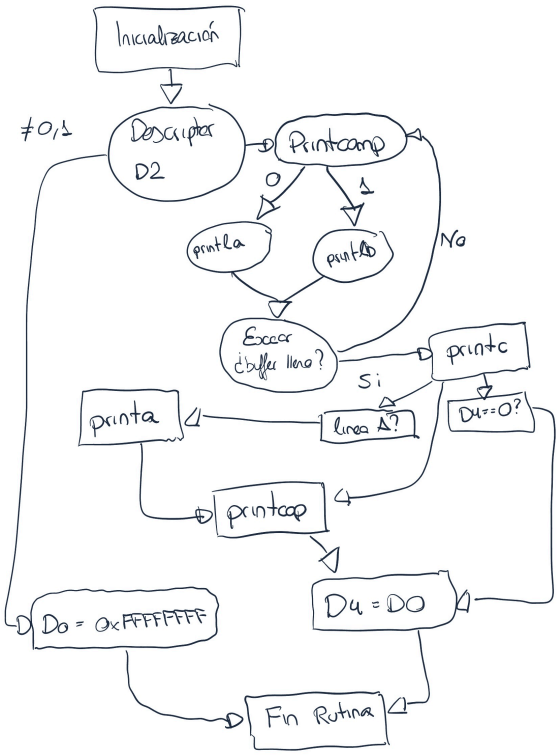
Printc

- Se comprueba el contador (registro D4)
- Si el contador es cero, no ha escrito, finalizo
- En caso contrario, se almacena el valor del registro de estado SR en un valor auxiliar (registro D5)
- Se inhiben la interrupciones
- Se comprueba si es linea A o B con el descriptor D2

- Si es la linea A se habilita las interrupciones de transmisión de A (IMR)
- Si es la linea B se habilita las interrupciones de transmisión de B (IMR)
- Se restaura el valor de SR original almacenado en D5

Se devuelve D4 en D0 el resultado.

El diagrama de flujo desarrollado aproximado sería el siguiente:



RTI

Esta es la rutina de tratamiento de interrupción, resultado de la ejecución de la secuencia de reconocimiento de instrucciones.

El funcionamiento desarrollado es el siguiente:

Pseudocódigo

- Se cargan los registros
- Se inicia un bucle (rtibucle) donde se comprueban los registros ISR y el registro imrcopia, que recordemos que es auxiliar debido a lo que se ha comentado en la introducción.
- En el bucle se hace el bit test para comprobar la interrupción
- Se comprueba primero el bit 0, en caso de que sea, transmisión linea A
- Se comprueba el bit 1, en caso de que sea, recepción linea A
- Se comprueba el bit 4, en caso de que sea, transmisión linea B
- Se comprueba el bit 5, en caso de que sea, recepción linea B

- **Bit 0 activado, transmisión A**
- Se carga el buffer interno de A
- Se llama a LEECAR
- Se realiza la comprobación del buffer, en caso de que esté vacio , se salta a inhibir interrupción
  - Se inicializa a cero el bit que corresponde a la transmisión de linea A en IMR
  - Se vuelve al bucle (rtibucle)
- Si la comprobación no da acierto, entonces se escribe lo que ha devuelto LEECAR en el buffer de transmisión de A
- Se vuelve al bucle (rtibucle)

- **Bit 1 activado, recepción A**
- Se carga el buffer de recepción de A
- Se llama a ESCCAR
- Se realiza la comprobación del buffer, en caso de que esté lleno , finaliza
- Se vuelve al bucle (rtibucle)

- **Bit 4 activado, transmisión B**
- Se carga el buffer interno de B
- Se llama a LEECAR
- Se realiza la comprobación del buffer, en caso de que esté vacio , se salta a inhibir interrupción
  - Se inicializa a cero el bit que corresponde a la transmisión de linea B en IMR
  - Se vuelve al bucle (rtibucle)
- Si la comprobación no da acierto, entonces se escribe lo que ha devuelto LEECAR en el buffer de transmisión de B
- Se vuelve al bucle (rtibucle)

- **Bit 5 activado, recepción B**
- Se carga el buffer de recepción de B
- Se llama a ESCCAR
- Se realiza la comprobación del buffer, en caso de que esté lleno , finaliza
- Se vuelve al bucle (rtibucle)

Programa principal de pruebas

Para la comprobación del correcto funcionamiento del proyecto se hizo uso del programa principal del que disponemos a partir de la página 75 del manual de la práctica. Para ver que operaba como se buscaba, se probó variando valores en los parámetros del programa.

```
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       * Tamano que se pasa como parametro
CONTC:      DC.W 0       * Contador de caracteres a imprimir
DESA:       EQU  0       * Descriptor línea A
DESB:       EQU  1       * Descriptor línea B
TAMBS:      EQU  30      * Tamano de bloque para SCAN
TAMBP:      EQU  7       * Tamano de bloque para PRINT


            * Manejadores de excepciones
INICIO:     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
                                         tamano
            MOVE.L #BUFFER,PARDIR        * Parametro BUFFER = comienzo
                                         del buffer
OTRAL:      MOVE.W PARTAM,-(A7)          * Tamano de bloque
            MOVE.W #DESA,-(A7)           * Puerto A
            MOVE.L PARDIR,-(A7)          * Direccion de lectura
ESPL:       BSR    SCAN
            ADD.L  #8,A7                 * Restablece la pila
            ADD.L  D0,PARDIR             * Calcula la nueva direccion de
lectura
            SUB.W  D0,PARTAM             * Actualiza el numero de
                                         caracteres leıdos
            BNE    OTRAL                 * Si no se han leıdo todas los
                                         caracteres
                                         * del bloque se vuelve a leer
```

```
            MOVE.W #TAMBS,CONTC         * Inicializa contador de
                                          caracteres a imprimir
            MOVE.L #BUFFER,PARDIR       * Parametro BUFFER = comienzo
                                          del buffer
OTRAE:      MOVE.W #TAMBP,PARTAM        * Tamano de escritura = Tamano
                                          de bloque
ESPE:       MOVE.W PARTAM,-(A7)         * Tamano de escritura
            MOVE.W #DESB,-(A7)          * Puerto B
            MOVE.L PARDIR,-(A7)         * Direccion de escritura
            BSR    PRINT
            ADD.L  #8,A7                * 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
```