# MEMORIA DEL PROYECTO ES\_INT.s DE ARQUITECTURA DE COMPUTADORES:

El objetivo del proyecto es que nos familiaricemos con la realización de Entrada/Salida en un periférico mediante interrupciones.   
Cabe mencionar que esta es la segunda vez que realizamos esta práctica, por lo que la gran parte del tiempo lo hemos invertido en corregir y realizar trazas del código ya escrito por nosotros el año pasado.

El proyecto se divide en la elaboración de las siguientes subrutinas:

* **INIT:** Subrutina que inicializa los dispositivos. Prepara las dos líneas por las que se van a recibir y transmitir caracteres y notificar las interrupciones.

Al principio parecía que esta subrutina era simple y la tuvimos escrita bastante rápido, de lo que no nos dimos cuenta y fue un error bastante grande fue de añadir una entrada, la de la RTI, esto causo varios problemas para testear el código ya que siempre nos fallaba en lo mismo. Después de escribir un correo y que nos aclararan las dudas, logramos corregirlo.

En escribir y corregir esta subrutina estuvimos alrededor de media hora ya que la teníamos escrita del proyecto del año pasado.

La teníamos planteada de la siguiente manera:

/////////////////////////////////////////////////////////////////////////////////////////////////

Inicilizar lineas A y B con parámetros

{

Dar acceso a reg. modo 1

8 bits por caracter

Desactivar el eco

Modo full duplex

Velocidad = 38400 bps tx y rx

}

Inicializaciones globales

{

Conjunto de veloc. 1

0x40 -> registro de vectores de interrupcion

Activar bits 1 y 5 para interr. RX

Actualiza copia del IMR

RTI -> primer vector de interrupcion ($100)

}

Inicializacion de buferes transmision y recepcion  
ini\_bufs();

Inicializacion de los contadores a cero y reseteo de A0

fin(RTS)

/////////////////////////////////////////////////////////////////////////////////////////////////

* **SCAN:** Subrutina en la que se lee de un dispositivo. Devuelve el bloque de caracteres previamente escrito por una de las dos líneas.

Al igual que con INIT, la teníamos escrita del año pasado, el problema era que el año pasado no nos paso muchas pruebas por lo que nos pusimos a arreglarla, el principal problema que tuvimos con esta subrutina y en general con todas las demás era como se trataba el marco de pila, so comprendíamos muy bien la instrucción MOVEM.L y esto nos generaba varios problemas al extraer los datos, para arreglar este problema sustituimos el MOVEM.L por varios MOVE.L de los registros que íbamos a utilizar en la subrutina.

Texto

Descripción generada automáticamente

Ilustración 1: solución al problema del marco de pila

Tras arreglar este problema, procedimos a probar la subrutina, y vimos que fallaba cuando se le pasaba un descriptor invalido, para simular esto, escribíamos directamente en la dirección de memoria correspondiente al descriptor un numero distinto de 0 o 1, después de trazar la subrutina, adelantamos un MOVE donde poníamos #FFFFFFFF en un registro que cuyo contenido posteriormente lo pasamos a D0 y añadimos un par de etiquetas para que D0 no se sobrescribiera.

Texto

Descripción generada automáticamente

Ilustración 2: Etiqueta nueva que salva D0

Con estos mismos cambios también arreglamos el resto de fallos que tenia la subrutina. El tiempo invertido en trazar la subrutina varias veces y corregir los errores que tenía no fue bastante, tardamos unas cinco horas.

La teníamos planteada de la siguiente manera:

/////////////////////////////////////////////////////////////////////////////////////////////////

meterPila(listaRegitros,RegPtroMarco)

elegirLineaXScan(descriptor){

if(descriptor==0){

D0 = 1

}

if(descriptor==1){

D0 = 1

}

else {D0=FFFFFFFF}

while(tamaño > 0 && contador!=tamaño) {

elegirLineaXScan(descriptor)

LEECAR(buffer)

if(d0 != FFFFFFFF){

ptro buffer ++

contador ++

}

else {

d0=0

fin(RTS)

}

}

D0 = contador

sacarPila()

return D0

fin(RTS)

/////////////////////////////////////////////////////////////////////////////////////////////////

* **PRINT:** Subrutina en la que se escribe en un dispositivo. Manda la escritura de el bloque de caracteres previamente escrito en una de las dos líneas.

Esta subrutina, también la teníamos del año pasado, aunque también nos tocó trazarla y corregirla. Hicimos el mismo cambio del marco de pila que en SCAN por el mismo motivo.

Texto

Descripción generada automáticamente

Ilustración 3: Solucion marco de pila

Después de comprobar que este cambio funcionaba y que ahora sí que se accedía bien a la pila procedimos a seguir trazando y corregir el primer error que aparecía en los resultados de las correcciones, el cual era el mismo que en SCAN, que no se devolvía #FFFFFFFF en D0 cuando de proporcionaba un descriptor erróneo, al tener el mismo problema, hicimos lo mismo para corregirlo, añadir una etiqueta que salvaba a D0 de sobrescribirse.

Texto

Descripción generada automáticamente

Ilustración 4: Solucion a etiqueta errónea

Luego corregimos el siguiente error que tenia que ver con el tamaño y las comprobaciones, si el tamaño del bloque que se enviaba a escribir era 0, no obteníamos el resultado esperado, esto lo arreglamos cambiando el orden de las comprobaciones y dejando solo las necesarias para saber en qué línea se va a escribir y dejando la comprobación del tamaño en el comienzo del bucle

Imagen que contiene Texto

Descripción generada automáticamente Texto

Descripción generada automáticamente

Ilustración :comparaciones Ilustración : comprobación tamaño (línea A)

Y a continuación procedimos a solucionar los problemas relacionados con la escritura y las interrupciones, esto fue lo mas costoso de corregir ya que tuvimos que reescribir el bucle y separarlo en dos uno para la línea A y otro para la B, con esto se nos arreglo el problema de la escritura en memoria pero no llegaba a escribir en ninguna de las dos líneas por como tratábamos el IMR, después de añadir un registro en el cual se guardaba el bit que íbamos a activar e introducirlo en IMR e IMRDUP conseguimos solucionar el problema de la escritura y de las interrupciones.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Ilustración 7: Correcion IMR

La corrección y la traza de esta subrutina fue lamas larga y tediosa, en ella ocupamos la mayor parte del tiempo dedicado a este proyecto tardamos unas 9 horas (varios días trabajando por la tarde).

La subrutina la planteamos de la siguiente manera:

/////////////////////////////////////////////////////////////////////////////////////////////////

Print (buffer, descriptor, tamaño)

meterPila(listaRegitros, RegPtroMarco){

crearMarcodePila(|listaRegistros| \* 4, RegptroMarco)

pushRegistros

Parametros(buffer, descriptor, tamaño) -> registros

}

elegirLineaX(descriptor){

if(descriptor==0){

D0 = 2

}

if(descriptor==1){

D0 = 3

}

else{

fin(RTS)

}

}

while(contador!=tamaño && D0!=FFFFFFFF)

linea = elegirLineaX(descriptor)

ESCCAR(D0, buffer++)

contador++

}

if (contador!=0){

habilitarTX(D0){

if (D0==0) bit num0=1 de IMRduplicado

if (D0==1) bit num4=1 de IMRduplicado

actualizar IMR

}

}

##contador == 0? -> no se activan interrupciones = branch fin Print

D0 = contador

sacarPila(){

popRegistros

destruccionMarcoPila(RegptroMarco)

}

return D0

fin(RTS)

/////////////////////////////////////////////////////////////////////////////////////////////////

* **RTI:** Subrutina que se ocupa del tratamiento de las interrupciones.

RTI trata los casos de interrupción en ambas líneas para la recepción y transferencia de caracteres. Nuestros principales errores fueron dos y tardamos poco en resolverlos gracias a los profesores que atienden el correo de dudas. El primero era que nuestra subrutina no era concurrente, lo que significa que en vez de comprobar si había más interrupciones antes de salir de RTI, salía y volvía a entrar en el antes del fetch de la siguiente instrucción. Algo que no afecta al funcionamiento pero si lo reduce considerablemente. El segundo error fue realizar un AND entre dos registros y comprobar en el que no guardaba el resultado de la operación. Ambos fallos muy triviales.

/////////////////////////////////////////////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////////////////////////////////////

En resolver y detectar estos dos errores nos llevó unas 3-4h de tiempo de trabajo (gracias otra vez al correo de dudas).