Práctica 3 - Ratón
Interrupciones y periféricos programables

Objetivos

- Desarrollar el hardware y los programas para manejar interrupciones concurrentes.
- Agregar periféricos a un sistema hardware dado.
- Utilizar periféricos programables (contador, temporizador y módulo de interrupciones).

Descripción General

Durante esta práctica se reescribirán las subrutinas de recepción serial desde un dispositivo PS2 para trabajar por interrupciones. De esta manera el programa podrá realizar otras tareas mientras se recibe una palabra.

Al finalizar la práctica se tendrá un temporizador comandado desde teclado o ratón. El temporizador se inicia en un valor prefijado (24,00 segundos) y se decrementa cada una décima de segundo hasta llegar a 0. También se puede modificar el valor inicial, arrancar, pausar y reiniciar mediante comandos desde un dispositivo de entrada PS2.

El programa que se debe escribir estará estructurado en subrutinas que hacen uso unas de otras y en rutinas de atención a interrupciones. Esto permitirá partir el problema en partes más sencillas de resolver y validar.

Hardware

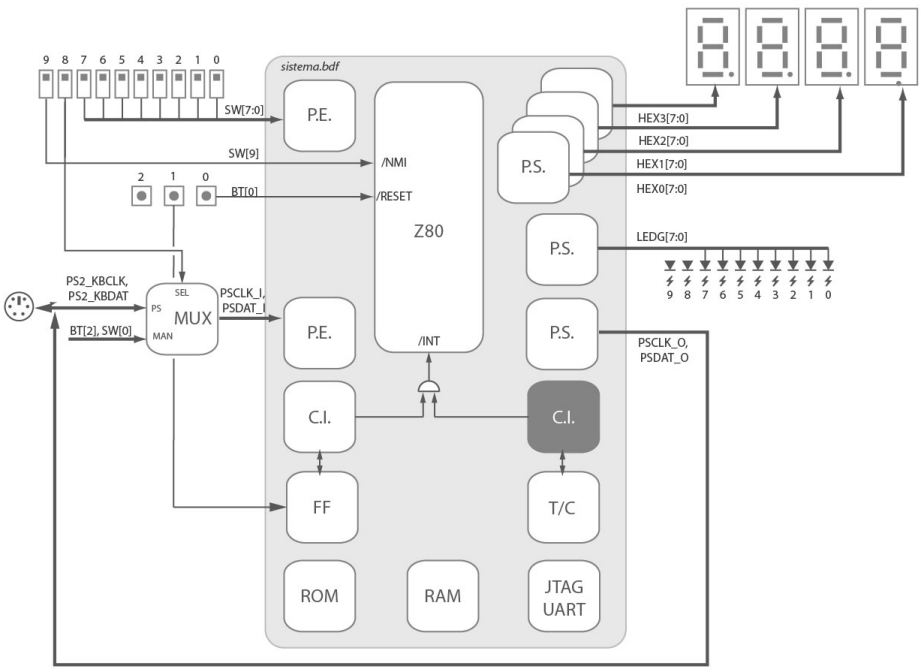

Figura 1: Descripción del hardware.

El hardware a utilizar en esta práctica consta de los mismos puertos de entrada/salida utilizados en la práctica anterior para conectarse a los LEDs, display 7 segmentos, llaves, pulsadores y puerto PS2. Notar que las direcciones elegidas para los puertos referentes al dispositivo PS2 pueden no coincidir con las utilizadas en la práctica 2.

Además se agregan algunos periféricos: controladores de interrupciones, timer y contador. El bloque T/C es un bloque Timer más un bloque Contador conectados en cascada.

Los bloques C.I. son controladores de interrupciones para usar en modo 2 e implementan el protocolo Daisy-Chain. La entrada INT del procesador puede ser activada o bien mediante el flanco de bajada de la señal PSCLK_I (proveniente del multiplexor) conectada a uno de los controladores o mediante el contador manejando otro controlador de interrupciones que deberá agregarse (Ver Figura 1).

El comportamiento exacto de estos bloques se describe en sus respectivos manuales de usuario que están disponibles en la página del curso. En color oscuro se muestra el periférico que deberá agregarse al sistema suministrado.

Tareas a realizar

a. Hardware

Analizar el sistema suministrado y completar las direcciones faltantes en la tabla de decodificación de puertos.

Rango direcciones	**Entrada**	**Salida**	**Observaciones**
0x80	SW[7:0]	HEX0[7:0]	
0x81		HEX1[7:0]	
0x82		HEX2[7:0]	
0x83		HEX3[7:0]	
0x84		LEDs[7:0]	
0x85	PS_DAT_I	PS_DAT_O	
0x86	PS_CLK_I	PS_CLK_O	
0x87		CL_PSCLK	Pulso de borrado de FF reloj PS2
	Timer	Timer	Rango de dir. ocupado por el timer
	Contador	Contador	Rango de dir. ocupado por el contador
	C. Int1	C. Int1	Rango de dir. ocupado por el controlador de interrupciones 1

Cuadro 1: Mapa de direcciones de puertos de entrada y salida.

b. Contador de interrupciones manuales

Subrutina *rutint_counter*

Descripción: Subrutina de atención a una interrupción que simplemente incrementa una variable en memoria y despliega en los dos dígitos menos significativos del display la cuenta. Para simplificar la rutina, el valor se despliega en hexadecimal utilizando las rutinas de las prácticas anteriores.

Como toda rutina de atención a interrupciones, deberá preservar todos los registros y habilitar interrupciones.

Las interrupciones se generarán utilizando el primer controlador de interrupciones mencionado en la sección Hardware.

Prueba: El objetivo de esta prueba es permitirnos entender y corroborar el funcionamiento de las interrupciones. Se debe realizar un programa que consista en la inicialización del sistema para trabajar con interrupciones en modo 2 (tanto el microprocesador como el controlador deben ser inicializados) y un bucle infinito que copie el estado de los switches a los LEDs para asegurarnos de que el sistema esté corriendo en forma correcta.

Durante la ejecución se deberá verificar que los switches encienden y apagan los LEDs correspondientes, y que cada vez que se presiona el pulsador que genera interrupciones, se incrementa la cuenta desplegada en los display menos significativos. Probarlo inicialmente seleccionando como entrada BUTTON[2]. Una vez validado seleccionar la entrada desde el dispositivo PS2, agregando al comienzo del programa de prueba las inicializaciones que sean necesarias (PSCLK_O y PSDAT_O a nivel alto, y también, llamar ***mouse_init*** antes de inicializar los controladores de interrupciones). ¿Cuánto debe incrementarse el contador al presionar el botón sin mover el mouse?

Nota: Para esta parte debe utilizarse el sistema suministrado sin modificaciones. Recién en la parte c) de la práctica se realizarán las modificaciones y se utilizarán en las siguientes partes.

c. Agregar segundo controlador de interrupciones

Modificar el hardware para agregar un segundo controlador de interrupciones manejado por el bloque contador. Este deberá tener más prioridad que el controlador que utiliza la entrada PSCLK_I.

Observar que en el sistema suministrado, el contador no solicita interrupciones, por ello la entrada INTA_n se encuentra conectada a GND. En la nueva configuración esta entrada deberá ser manejada por el nuevo controlador de interrupciones.

d. Interrupciones periódicas

d.1. Versión simple: LEDs intermitentes

Se deberá agregar al programa principal la inicialización del nuevo controlador de interrupciones y de los bloques timer y contador para que interrumpan cada 1 décima de segundo con un error menor a 1 milisegundo (incluir cálculo en el informe). La rutina de atención para esta interrupción deberá complementar el valor de una variable y escribirla en el dígito más significativo (HEX3) del display de 7 segmentos.

En el circuito puede verse además que el led 9 está conectado a un FF que cambia su salida cada vez que se genera un pulso en ZC del contador, esto puede utilizarse para ver si efectivamente el contador está llegando a 0 cada 1 décima de segundo.

Nota: Como se mencionó anteriormente y para todos los programas en adelante, la inicialización del ratón mediante ***mouse_init*** debe realizarse antes de inicializar los bloques controladores de interrupciones.

Prueba: Con las modificaciones realizadas al programa de parte b) se observará que el programa principal y ambas interrupciones funcionan en forma independiente: los LEDs con la copia de la entrada SW, HEX1 y HEX0 con la cuenta de interrupciones producidas por PSCLK_I y HEX3 conmutando entre encendido y apagado cada una décima de segundo.

d.2. Versión completa: subrutina *rutint_timer*

Se sustituye la subrutina de atención a la interrupción del segundo controlador por la subrutina ***rutint_timer*** que, si la variable PAUSA vale 0x00, decrementa variables en memoria que llevan la cuenta de segundos y décimas y las presenta en los 4 dígitos del display.

Para el decremento y actualización de display deben utilizarse las subrutinas ***decreloj*** y ***despreloj*** implementadas en la práctica 1.

PAUSA es actualizada desde fuera de ***rutint_timer*** y vale 0xFF si se debe pausar la cuenta. En el recuadro se da un posible seudocódigo de esta subrutina.

```
rutint_timer:
   habilito int.
   preservo registros
   if !pausa then
     decreloj()
     despreloj()
   endif
   restauro registros
end rutint_timer
```

Prueba: Agregar al loop del programa principal de la parte anterior la actualización de la variable PAUSA de acuerdo al estado del switch 0. Mientras PAUSA es cero conviene evitar generar interrupciones con PSCLK_I de modo de no tener conflictos en el uso del display por parte de las dos interrupciones.

e. Leer un dato serie

Subrutinas *get_ps2_nb* (suministrada) y *ps2clk_isr*

Descripción: Para la recepción serie se utiliza por un lado la subrutina ***get_ps2_nb*** que se suministra junto con la letra de la práctica en el archivo *get_ps2_nb.s* y por otro lado la subrutina ***ps2clk_isr*** de atención a la interrupción generada por el flanco en PSCLK_I que deberán escribir. La subrutina ***ps2clk_isr*** debe ir procesando la recepción de un dato serie de a un flanco en cada interrupción. Se utilizarán variables en memoria para almacenar el estado de avance de la recepción (es decir el dato parcialmente recibido y cuántos flancos se llevan procesados) entre interrupciones sucesivas. La comunicación de ***ps2clk_isr*** con ***get_ps2_nb*** también se realiza mediante variables en memoria, las cuales sirven para indicar si hay o no un nuevo dato recibido y en caso afirmativo almacenar el valor del mismo. Dichas variables (rx_done, rx_byte) son utilizadas por ***get_ps2_nb*** y deben ser manejadas en forma coherente por ***ps2clk_isr***.

Observar que a diferencia de la subrutina ***get_ps2*** usada en la práctica 2, la subrutina ***get_ps2_nb*** suministrada siempre retorna inmediatamente sin quedar bloqueada a la espera de la recepción de un dato. Devuelve como resultado una indicación de si hay dato disponible o no, y en caso afirmativo devuelve también el dato recibido. En esta nueva versión el valor de paridad recibido se descarta.

Se pide analizar y comprender el funcionamiento de la subrutina ***get_ps2_nb*** suministrada y escribir una subrutina ***ps2clk_isr*** que se encargue de la recepción y se comunique correctamente con ***get_ps2_nb***.

Parámetros: ***get_ps2_nb*** devuelve si hay un dato disponible o no en el flag Z (prendido si hay dato, apagado en caso contrario) y en caso de haber dato disponible lo devuelve en el registro A.

Prueba: El programa de prueba debe ser un bucle infinito que invoque ***get_ps2_nb*** y que, en caso que ***get_ps2_nb*** devuelva un nuevo dato despliegue en los dos dígitos menos significativos del display 7 segmentos el último dato recibido. Además, el programa debe mostrar en los dos dígitos más significativos el penúltimo dato y en los 8 LEDs (LEDG[7:0]) el antepenúltimo dato recibido. Igual que en el caso anterior se sugiere utilizar las subrutinas de conversión a 7 segmentos de la práctica 1. En los LEDs el dato se despliega sin convertir. Notar que el loop de prueba es similar al usado en la práctica 2 salvo que la actualización de salidas debe hacerse condicional con el valor del flag Z devuelto por ***get_ps2_nb***. Para evitar interferencias en el uso de los displays se sugiere no

habilitar la interrupción del timer-contador.

f. Procesar datos recibidos

Subrutina *get_packet_nb*

Descripción: Deben escribir la subrutina ***get_packet_nb***. Al igual que en el caso anterior, la nueva subrutina ***get_packet_nb*** es "no bloqueante", es decir siempre retorna inmediatamente sin quedar bloqueada a la espera de la recepción. Devuelve como resultado una indicación de si hay un paquete disponible o no, y en caso afirmativo devuelve el contenido del paquete en memoria.

Esta subrutina deberá invocar a ***get_ps2_nb*** y usará variables en memoria para llevar la cuenta del avance en la recepción del paquete entre invocaciones sucesivas y para memorizar el estado anterior de los botones. Respecto a la subrutina ***get_packet*** de la práctica anterior se agrega un byte que indica si hubo una transición de 0 a 1 en el estado de los botones.

Se debe preservar todos los registros excepto el acumulador y las banderas.

Parámetros: Recibe en IX la dirección base de comienzo de la estructura de datos en memoria detallada en el cuadro 2. Devuelve si hay un paquete disponible o no en el flag Z (prendido si hay paquete, apagado en caso contrario). Si hay un paquete disponible lo devuelve en el lugar de memoria a partir de la dirección base.

(base)	Primer byte recibido conteniendo banderas de overflow, signos y estado de botones
(base+1)	Byte bajo de desplazamiento X representado en complemento a 2 de 16 bits
(base+2)	Byte alto de desplazamiento X representado en complemento a 2 de 16 bits
(base+3)	Byte bajo de desplazamiento Y representado en complemento a 2 de 16 bits
(base+4)	Byte alto de desplazamiento Y representado en complemento a 2 de 16 bits
(base+5)	Click: banderas que indican transición de no presionado a presionado en el estado de botones (bit igual a 1 indica que hubo transición de 0 a 1 en el botón correspondiente). Solo tienen sentido los bits 2,1 y 0 que corresponden a los botones central, derecho e izquierdo respectivamente. Se calcula fácilmente como el AND entre el valor anterior negado y el actual del campo (base+0).

Cuadro 2: Descripción de parámetros recibidos por la subrutina.

Prueba: El programa de prueba reserva espacio en memoria para los 6 bytes y luego queda en un bucle infinito invocando ***get_packet_nb*** y actualizando, cuando se recibe un nuevo paquete, el estado de los LEDs y dígitos 7 segmentos. Los datos a mostrar dependen del valor de SW[0]: si SW[0]=1 se muestra el primer byte (base+0) y el desplazamiento en X; si SW[0]=0 se muestra el último byte (base+5) y el desplazamiento en Y. El primer y último byte se muestran en los LEDs y los desplazamientos en los 4 dígitos 7 segmentos.

g. Temporizador comandado desde ratón

En esta parte se implementará un temporizador comandado desde ratón. Se suministra el código fuente del programa principal para una primera versión simplificada. Se pide analizar el código suministrado en el archivo *temporizador_mouse_template.s*, integrarlo con las subrutinas probadas en las partes anteriores y probar el funcionamiento del temporizador simple. Luego se debe agregar los estados faltantes y probar el funcionamiento del temporizador completo.

Funcionamiento: El temporizador responde a dos comandos: ***pp*** (pause/play) y ***restart*** enviados desde el ratón (ver Cuadro 3). La versión simple suministrada evoluciona entre los estados ***inicial***, ***contando*** y ***fin***, los tres estados dentro del recuadro en línea punteada en el diagrama de estados de la Figura 2. Los cambios de estado son provocados por la llegada de un comando o porque el temporizador llega a cero. Para decrementar y desplegar el valor del temporizador en el estado ***contando*** se usa la interrupción generada por el conjunto Timer-Contador con la subrutina ***rutint_ timer*** ya probada. En el estado ***inicial*** se debe llevar la cuenta a 24.00 segundos y mantener al temporizador pausado. En el estado ***contando*** se habilita la cuenta con la variable PAUSA. Finalmente en el estado ***fin*** se debe pausar la cuenta y encender 4 LEDs (LEDs[3..0]).

restart	Right click (bit 1 del byte 5 del paquete)
pp	Left click (bit 0 del byte 5 del paquete)

Cuadro 3: Codificación comandos.

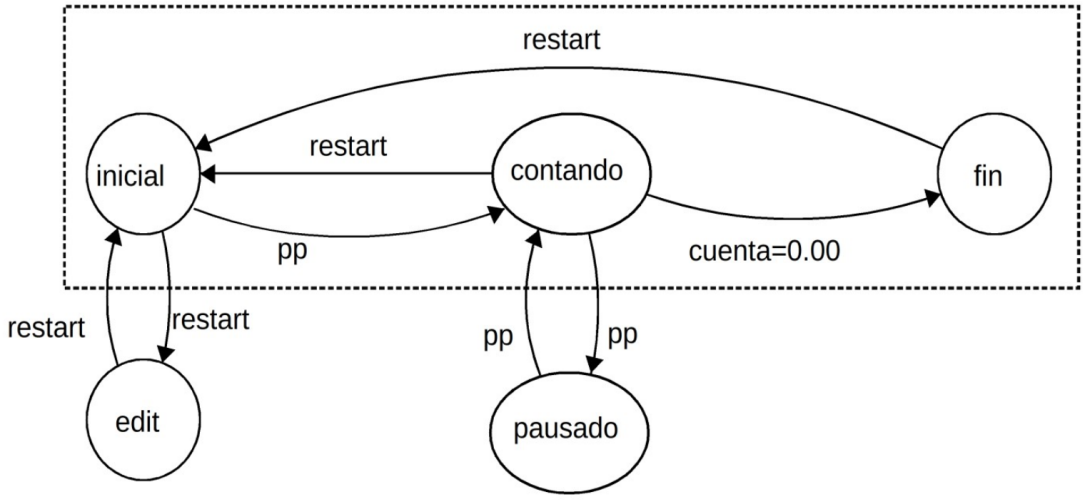

Figura 2: Diagrama de estados del temporizador.

En una primera etapa se debe analizar y entender el programa principal suministrado, integrarlo con las subrutinas probadas en las partes anteriores y hacer funcionar la versión simple con los estados ***inicial***, ***contando*** y ***fin***.

En la segunda etapa se deben agregar las funcionalidades de los estados ***pausado*** y ***edit***. En el estado ***pausado*** simplemente se detiene la cuenta quedando fijo en el display el valor que tenía al momento de entrar en pausa. En el estado ***edit*** se habilita el ingreso del valor de arranque de los segundos mediante el ratón y se encienden los LEDs[7..4]. El valor de las centésimas se mantiene sin modificar en 0.

Al entrar en modo edición, los display mostrarán 00:00 y se esperará el ingreso del nuevo valor de segundos mediante el ratón. Los desplazamientos horizontales se usan para incrementar o decrementar una cuenta acumulada similar a lo hecho en la práctica 2. Si se recibe un ***restart***, se debe actualizar la variable que lleva la cuenta de segundos y retornar al estado ***inicial*** con el nuevo valor inicial del temporizador. (Ver Figura 3)

Al igual que en la práctica 2, para disminuir la sensibilidad ante los movimientos del ratón, el valor acumulado de desplazamiento debe dividirse por 2^N con la subrutina ***barrel_ shift_ right*** antes de desplegarlo en los display y de almacenarlo en la variable de segundos del reloj. Invocar la subrutina pasando el valor de N determinado en la práctica 2 tomado de una constante y no de los switches.

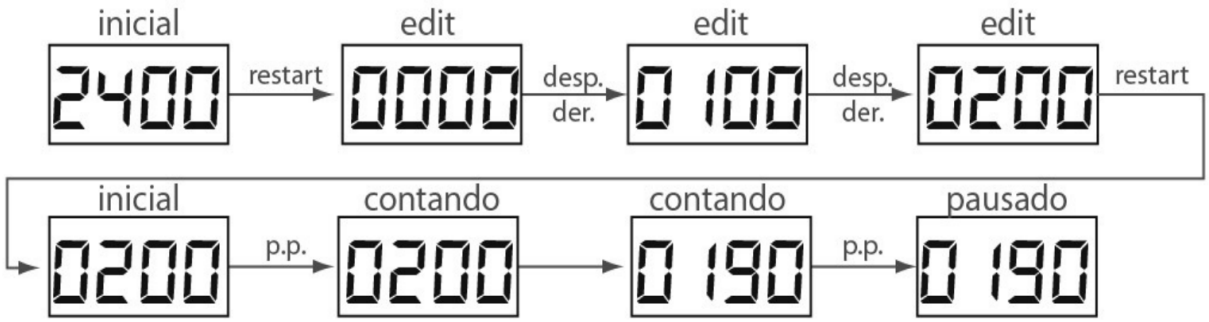

Figura 3: Ejemplos

h. Ejercicio

Se debe completar el diagrama de tiempos mostrado en la Figura 4. Las primeras tres señales son las del controlador de interrupciones encargado de manejar la interrupción proveniente de la señal PS2CLK_I y las segundas tres señales son las del controlador de interrupciones asociado a la interrupción generada por el bloque contador. Las líneas verticales indican una serie de eventos externos a los controladores. En base a estos, se debe dibujar cómo se modifican las señales indicadas. El primer evento, consiste en que el bloque contador finalizó su cuenta con lo cual solicita una interrupción a su controlador. El segundo evento se da cuando llega un flanco en PS2CLK_I (p. ej. se recibe un bit en la comunicación serie). Por último se marca la ejecución de dos RETI por parte del microprocesador. El diagrama resuelto debe ser incluido en el informe indicando también a qué rutina de atención a interrupción se encuentra asociado cada RETI.

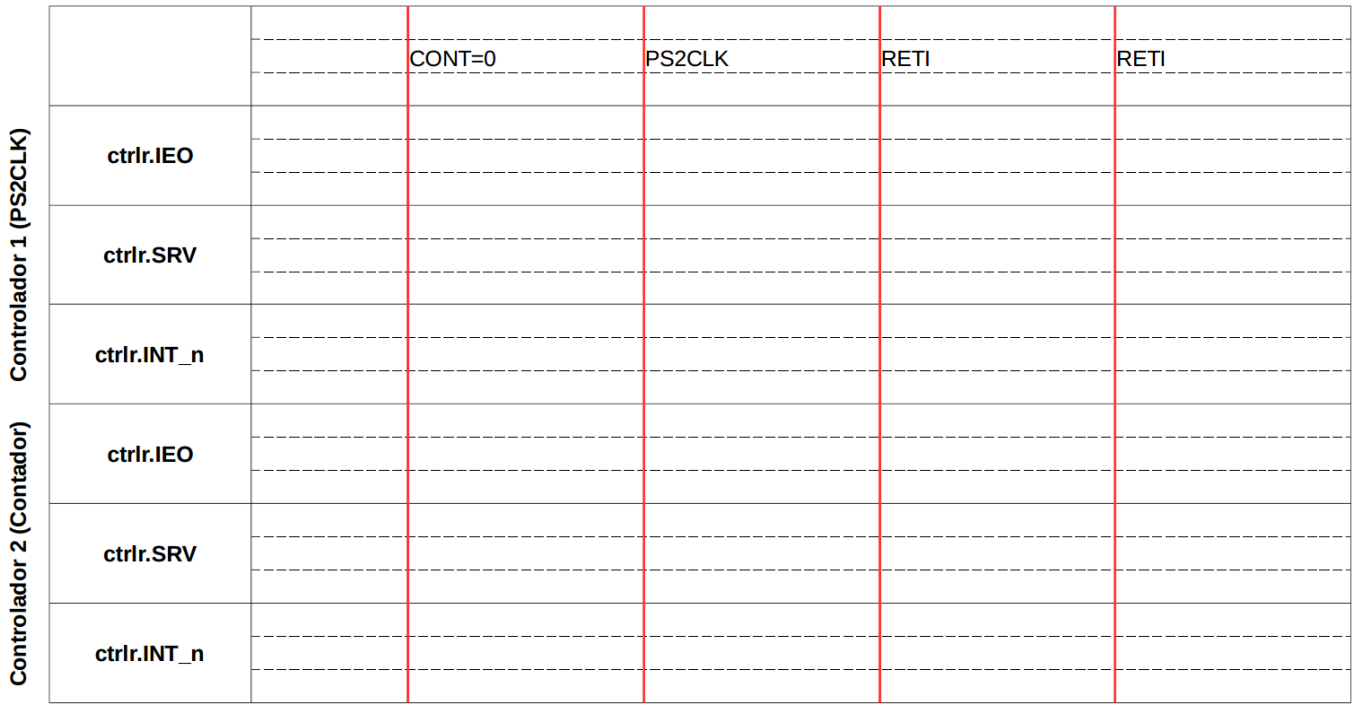

Figura 4: Ejercicio

Informe

La entrega consistirá de:

(I) Informe: archivo *dd-hh-mm-n-informe.pdf* (donde dd-hh-mm-n son el día, hora y número que identifican al grupo, p. ej.: ma-14-00-3). Este archivo deberá contener:

- Mapa de entrada/salida con puertos originales y puertos agregados.
- El pseudocódigo o el diagrama de flujo de las subrutinas y programas de prueba.
- El código assembler de las subrutinas y programas de prueba
- Cálculo para la inicialización de los bloques timer y contador, para cumplir con los requerimientos de tiempo de la parte d).
- Esquemático de la conexión del segundo controlador de interrupciones agregado en la parte c) y de las señales de decodificación correspondientes.
- El diagrama de tiempos del ejercicio planteado en la parte h).

(II) Programas: archivo comprimido en formato zip (*dd-hh-mm-n-fuentes.zip*) conteniendo los siguientes archivos:

- El código de todas las subrutinas (*subrutinas.s*)
- Programas de prueba (*prueba_b.s* a *prueba_g.s*) para cada una de las subrutinas que incluyen el archivo anterior mediante la directiva .include.

(III) Hardware modificado: archivo comprimido en formato zip (*dd-hh-mm-n-hardware.zip*) con los siguientes archivos del diseño realizado por el grupo.

- *sistema.bdf*
- *lab_intup.sof*
- *lab_intup.qsf*

Se entregará antes del **lunes 17 de junio a las 13:00 hs** un archivo formato .zip con todos los ítems mediante la tarea correspondiente en EVA.

Cada estudiante **deberá llevar registro de las horas dedicadas a la práctica**. Se les solicitará que ingresen esa información a través de la página del curso.

El día de la evaluación **el grupo deberá presentarse 10 minutos antes de la hora establecida** en el laboratorio de software del instituto de Ingeniería Eléctrica.

Además deberán traer el KIT DE0-LAB y un "pen drive usb" con todos archivos de los proyectos indicados y las simulaciones realizadas, si se desea usar las pc de la facultad. Si utilizaron una laptop para hacer la tarea se recomienda traer la misma laptop para la defensa. En ambos casos probar que todos los programas funcionan antes de la práctica.