Entrada/Salida

Organización del Computador I

Segundo Cuatrimestre 2023

Departamento de Computación - FCEyN
UBA

Introducción

Hasta ahora vimos:

- Representación de números enteros y decimales
- Circuitos combinatorios y secuenciales
- Un poco de programación en assembler de $\mathrm{ORGA1}$
- Arquitectura y microarquitectura

Pero...

¿Podríamos conectarnos con el mundo exterior?

¿Tendría algún sentido?

¿Y cómo lo haríamos?

Dispositivos de E/S

¡Interactuamos con el mundo exterior a través de los dispositivos de E/S!

- **Entrada:** teclado, mousepad, lector de huella, webcam, etc.
- **Salida:** monitor, placa de sonido, impresora, etc.
- **Entrada/Salida:** disco rígido, router, usb flash, etc.

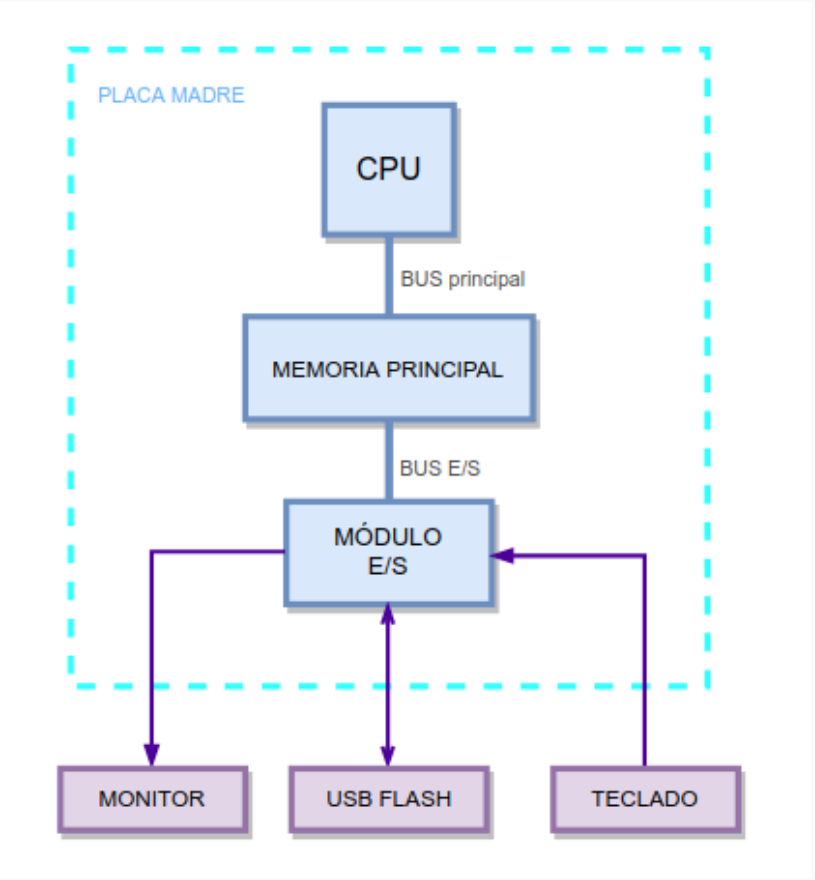
¿Cómo intercambiamos información con un dispositivo de E/S?

Intercambio de información con un dispositivo de E/S

Cada dispositivo de E/S tiene sus propios registros en los que la CPU puede leer o escribir datos.

Tipos de registro:

- **Lectura** (para dispositivos de Entrada)
- **Escritura** (para dispositivos de Salida)
- **Lectura/Escritura** (para E/S)



Métodos de acceso a los registros

¿Qué instrucciones nos permiten acceder a estos registros?

Hay dos maneras de referirse a ellos:

- E/S independiente (instrucciones especiales: IN y OUT).
  Espacio de direcciones independiente
- E/S mapeado a direcciones de memoria.
  Direcciones de memoria principal reservadas para E/S (Utilizaremos este en la práctica)

Es por eso que ORGA1 reserva las direcciones de memoria `0xFFF0` a `0xFFFF` para los dispositivos de E/S.

Métodos de control de E/S

¿Y cómo controlamos estos dispositivos? Primero pensemos un poco en problemas de la vida diaria:

Ir a beldía por el control del proyector

Ir al baño

Métodos de control de E/S

Ir a beldía por el control del proyector

- Tengo que revisar constantemente si llegó o no el control para el proyector
- No necesito moverme de donde estoy o irme de la clase
- No me doy cuenta que el control llegó hasta que pregunto por este.

Métodos de control de E/S

Ir al baño

- Tengo un aviso o interrupción cuando necesito ir al baño
- Necesito levantarme e ir físicamente al baño, dejando lo que estoy haciendo
- Al volver tengo que saber lo que estaba haciendo!

Métodos de control de E/S

Volviendo a Orga1, podemos controlar la interacción con los dispositivos de E/S de dos formas:

- **Polling** (como el ejemplo de esperar a que llegue el control)
- **Interrupciones** (como el ejemplo de ir al baño)

Métodos de control de E/S: Polling

- El sistema posee al menos un registro exclusivo para cada dispositivo, la CPU los debe monitorear constantemente.
- Cuando la CPU *detecta* que en algún registro hay un dato de relevancia, entonces actúa según corresponda.
- Es el método más lento, pero que tiene una complejidad baja en cuanto al hardware. El Módulo de E/S se conecta solo con la Memoria, no tiene conexión con la CPU.

Métodos de control de E/S: Interrupciones

- La CPU no está constantemente monitoreando los registros.
- Cada dispositivo interrumpe a la CPU al enviar una señal de interrupción.
- Es por eso que hay una conexión entre el Módulo E/S y la CPU.
- La CPU debe interrumpir su tarea para atender al dispositivo. Para ello guarda la información del contexto actual, carga la dirección de la rutina de atención de interrupción, la ejecuta y al terminar restaura el estado anterior.

¿Y si quiero mover grandes volúmenes de datos?

- El Módulo de DMA controla la transferencia de datos entre el dispositivo E/S y la memoria sin pasar por el procesador.
- Puede avisarle al procesador que la transferencia está lista por medio de una interrupción, o se puede hacer polling sobre el dispositivo DMA para averiguarlo.
- Este controlador comparte el mismo BUS de datos con el que la CPU se comunica con la Memoria y además se conecta con la CPU para el envío de diferentes señales.

Estructura de manejo de E/S utilizando polling

¿En qué casos utilizamos polling?

- Se tiene un sensor que mide un valor de la realidad. Por ejemplo: temperatura, humedad, cantidad de veces que un objeto fue detectado por el sensor.
- Un sensor que no tiene memoria ni capacidad de interrumpir.
- La CPU no tiene otra cosa para hacer o disponemos de múltiples unidades de procesamiento (múltiples cores).
- Se conoce que el sensor muestra valores con cierta frecuencia. Los valores que no se leen en ese intervalo de tiempo, se pierden.

Estructura típica

Enunciado típico En el siguiente programa, monitoreamos la temperatura y cantidad de mediciones que nos da un sensor mapeado a la dirección 0xFFF0. Los valores posibles de temperatura van en el rango 0x00 y 0xFF (0-255 decimal).

- Realizar el mapeo de registros.
- Dar una rutina en lenguaje ensamblador para acumular la suma de las temperaturas medidas y contar la cantidad de mediciones tomadas.

Mapeo de registros en memoria

Registro de lectura	Dirección 0xFFF0	Valores posibles: 0x00 - 0xFF

Programa

```
comienzaPrograma:   MOV R0, 0x0000      ;En R0 tengo las temperaturas
                    MOV R1, 0x0000      ;En R1 cuento las mediciones
GuardoMedición:     ADD R0, [0xFFF0]    ;Sumo el valor de temperatura
                    INC R1              ;Cantidad de mediciones ++
                    JMP GuardoMedición  ;Continúo guardando mediciones
```

¿Cómo venimos?

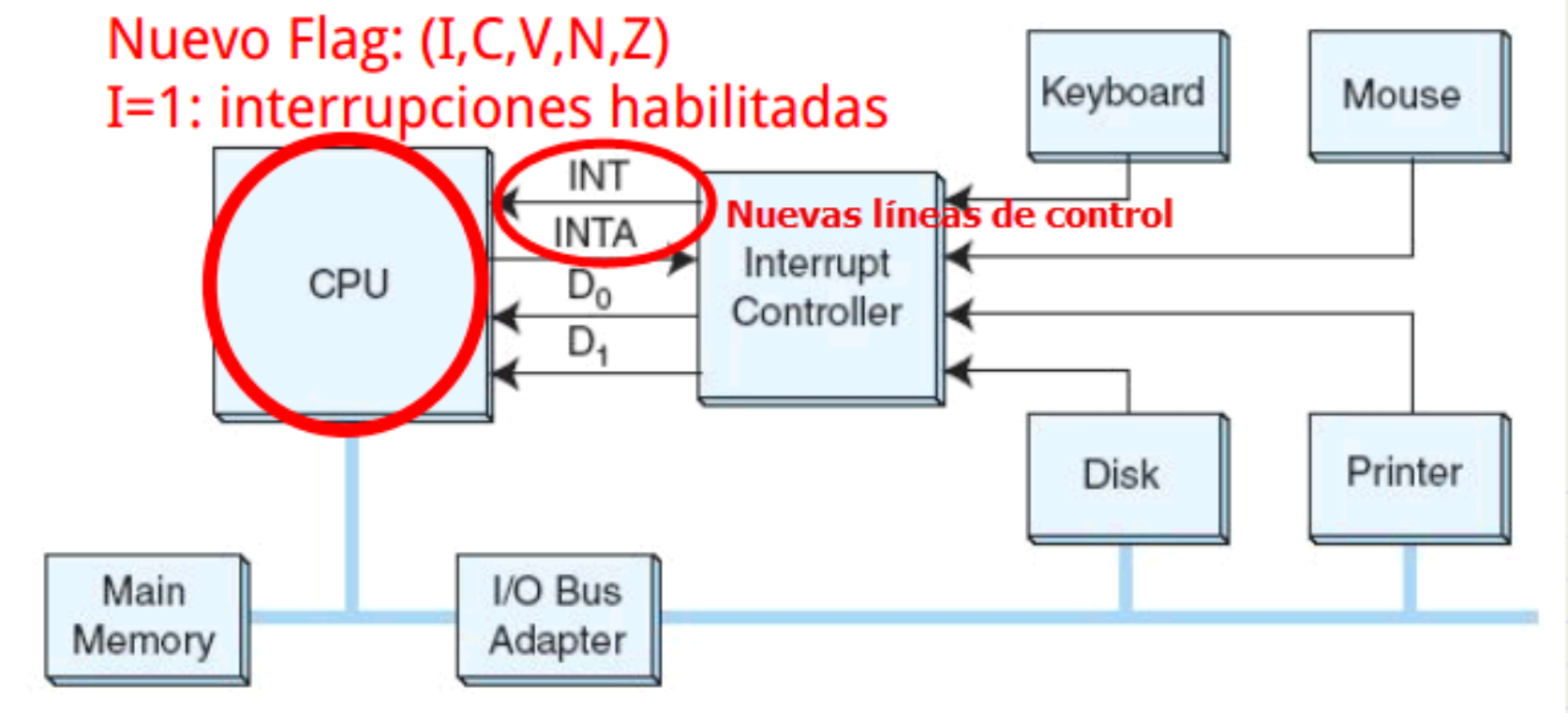
Interrupciones

El procesador ORGA1i

El procesador ORGA1I es una extensión de ORGA1 con la capacidad de atender una única interrupción (enmascarable) de un dispositivo de E/S.

- **Nuevas señales:**
  - Entrada: `INT` (pedido de interrupción)
  - Salida: `INTA` (reconocimiento de interrupción)
- **Nuevo flag:** `I`, que indica si el procesador puede ser interrumpido.
- **Nuevo registro:** `PSW (Program Status Word)`, donde se almacenan los flags.
- **Nuevas instrucciones:**
  - `CLI` y `STI`, que setean el flag `I` en 0 y 1 respectivamente.
  - `PUSH Ri`, cuyo efecto es `[SP]=Ri` y luego `SP=SP-1`.
  - `POP Ri`, cuyo efecto es `SP=SP+1` y luego `Ri=[SP]`.
  - `IRET`, cuyo efecto es `PC=[SP+1]`, `PSW=[SP+2]` y `SP=SP+2`.
- **Nueva dirección reservada:** `0x0000`, donde se indica la dirección de la rutina de atención de las interrupciones.

El procesador ORGA1i



¡Recuerden que esto está en la clase teórica!

¿Qué pasa cuando interrumpen a la CPU?

En el caso del procesador $\mathrm{ORGA1I}$, si el dispositivo de E/S activa la señal de interrupción y el flag `I` vale 1, este termina de ejecutar la instrucción en curso y realiza los siguientes pasos **atómicamente**:

- Asigna [SP] = PSW y decrementa SP.
- Asigna [SP] = PC y decrementa SP.
- Realiza `I` $\leftarrow 0$ para evitar que el procesador vuelva ser interrumpido.
- Asigna PC = [0x0000].
- Activa la señal INTA para indicarle al dispositivo que atenderá su pedido.

Seguidamente comienza a ejecutarse la rutina de atención de la interrupción propiamente dicha.

Estructura de manejo de E/S utilizando interrupciones

¿En qué casos utilizamos interrupciones?

- Tenemos un dispositivo que puede enviar un mensaje de interrupción.
- Típicamente tiene asociado un controlador de interrupciones.
- El protocolo de comunicación está bien definido, nos interrumpen cada vez que hay un nuevo valor disponible.

Estructura típica

Enunciado típico se quiere guardar los valores numéricos que genera un usuario a través de un teclado, solo se recibirán valores del 0 al 9 y se guardarán tan solo los últimos 4 dígitos presionados. Los distintos valores ingresados por el usuario estarán mapeados a memoria en el registro INPUT_USER. Se desean acumular los últimos 4 dígitos presionados en el registro R3.

Mapeo de registros en memoria INPUT_USER | 0xFFF0 | Valores posibles: 0x0 - 0x9

Rutina de atención a la interrupción

```
rut_at_int:   PUSH R0 ; Guardamos el valor de R0
              MOV R0, [0xFFF0] ; Obtenemos el nuevo valor
              SHL R3, 4 ; Movemos 4 bits a izquierda para acumular el nuevo valor
              ADD R3, R0 ; Sumamos el valor de R0 y acumulamos
              POP R0 ; Restauramos el valor original de R0
              IRET
```

Ejercicios

Ejercicio 1

Se tiene una computadora $\mathrm{ORGA1I}$ para monitorear el estado de una montaña rusa y actuar acorde a cada uno. Para ello, cuenta con dos dispositivos de E/S que actúan como sensores, una alarma y otro que efectúa una parada de emergencia.

Cada sensor posee un registro de E/S que reporta lo siguiente:

- Velocidad: Mide la velocidad del carrito de la montaña rusa (VEL_STATUS).
- Frenos: Mide el estado de los frenos (BR_STATUS).

Las etiquetas MAX_SPEED y MIN_BRAKES son constantes de 16 bits.

La alarma posee un registro ALARMA con el que se activan las alarmas al setearse algún bit en 1. El bit menos significativo representa la alarma de velocidad, que indica que el carrito está yendo muy rápido. El segundo bit representa el estado de los frenos comunes e indica que hubo un problema con ellos y se activarán los frenos de emergencia. Las alarmas se apagan manualmente por fuera del sistema.

El dispositivo de frenos de emergencia posee un registro de E/S de escritura (FRENOS_EM) que los activa al detectarse un problema con los frenos comunes. Para ello deben setearse todos sus bits en 1.

Ejercicio 1 (cont.)

1. Mapear los registros de E/S a direcciones de E/S de ORGA1I.
2. Realizar el código para sensar y activar las alarmas correspondientes.
3. Suponiendo que cada ciclo de instrucción del programa demora 3 ms y los valores máximos nunca son alcanzados ¿Cuál es la frecuencia (en Hz) de muestreo (lectura) de los sensores? ¿Y si todos los sensores sobrepasan los máximos?

Solución 1.a)

Mapeo de registros:

- VEL_STATUS $\mapsto$ 0xFFF0
- BR_STATUS $\mapsto$ 0xFFF1
- ALARMA $\mapsto$ 0xFFF2
- FRENOS_EM $\mapsto$ 0xFFF3

Solución 1.b)

```
MIN_BRAKES:    DW ....
MAX_SPEED:     DW ....
sensarVel:     CMP [0xFFF0], [MAX_SPEED] ;¿se alcanzó velocidad máxima?
               JL sensaFrenos
               OR [0xFFF2], 0x0001
sensaFrenos:   CMP [0xFFF1], [MIN_BRAKES] ;¿hay problema con los frenos?
               JG sensaVel
               MOV [0xFFF3], 0xFFFF
               OR [0xFFF2], 0x0002
               JMP sensaVel
```

Solución 1.c)

De no alcanzarse los valores máximos, cada iteración ejecutará 4 instrucciones. Por lo tanto, cada iteración tardará $4 * 3$ ms.
Como sólo podemos realizar una lectura por iteración, cada señal será leída cada $4 * 3$ ms.
Finalmente (y sabiendo que 1000 ms $= 1$ s), la frecuencia de muestreo será de $\frac{1000}{4*3} = 83,33$ Hz.

En cambio, de sobrepasarse los valores máximos, cada iteración ejecutaría 8 instrucciones, tardando $8 * 3$ ms.
Cada señal se leería cada $8 * 3$ ms, dando una frecuencia de muestreo de $\frac{1000}{8*3} = 41,66$ Hz.

Ejercicio 2

El dueño de la montaña rusa decidió invertir en un nuevo sensor para los frenos. Este nuevo sensor solicita una interrupción al detectar un inconveniente con los frenos comunes.

a) Escribir la rutina de atención de la interrupción del sensor de frenos.
b) Modificar el programa presentado para aprovechar esta característica y aumentar la frecuencia de muestreo.
c) Calcular la nueva frecuencia de muestreo para el sensor de velocidad.
d) ¿Cuál sería el estado de la memoria al cargar el programa si se pide que la rutina principal se cargue a partir de la posición 0xAAAA y la de atención de la interrupción 20 posiciones antes?

Solución 2)

Rutina de atención de la interrupción

```
rut_at_int:  MOV [0xFFF3], 0xFFFF
             OR [0xFFF2], 0x0002
             IRET
```

Modificación

```
sensaVel:  CMP [0xFFF0], [MAX_SPEED] ;¿se alcanzó la velocidad máxima?
           JL sensaVel
           OR [0xFFF2], 0x0001
           JMP sensaVel
```

Solución 2)

Muestreo

De no alcanzarse el valor máximo, cada iteración ejecutaría 2 instrucciones, resultando en una frecuencia de muestreo de $\frac{1000}{2*3} = 166,66$ Hz.
En cambio, de sobrepasarse este valor, cada iteración ejecutaría 4 instrucciones, dando una frecuencia de $\frac{1000}{4*3} = 83,33$ Hz.

Estado de la memoria

La rutina principal se cargaría a partir de la dirección 0xAAAA y la de atención de la interrupción a partir de 0xAA96.
Consecuentemente, en la posición 0x0000 cargaríamos 0xAA96, la dirección en que se cargó la rutina de atención.

Ejercicio 3

Se cuenta con un dispositivo capaz de almacenar agua en un tanque y dejarla pasar a través de una exclusa cuando su operario lo disponga. El sistema es de llenado automático con una luz que avisa cuando el tanque está lleno. Se desea controlarlo con un procesador ORGA1I, notando que posee:

- un `BOTÓN` presionable por el usuario. Su estado se ve reflejado en el registro `VAL_BOTON`: de estar presionado vale `0xB010`, si no `0x0000`.
- un `LED` que debe encenderse cuando el tanque esté lleno para indicar que el sistema está listo para volver a utilizarse. Su registro es `LED_STATE`, para encenderlo se debe escribir `0xFFFF` y para apagarlo, `0x0000`.
- una `EXCLUSA` que permite dejar pasar el agua. Para abrirla se debe escribir continuamente el valor `0xAB1E` en el registro `SALIDA`. En cambio, para cerrarla se debe escribir `0xCE11`.
- un `NIVEL` que produce una interrupción cuando el agua sobrepasa los límites permitidos. Al producirse la interrupción, se escribe `0xA17A` en el registro `NIV_AGUA` si el agua superó el límite superior, y `0xBA1A` si superó el inferior.
- una `CANILLA` que deja entrar agua al tanque. La abrimos escribiendo `0xADE7` en el registro `CANILLA_ST`, y la cerramos escribiendo `0x00FF`.

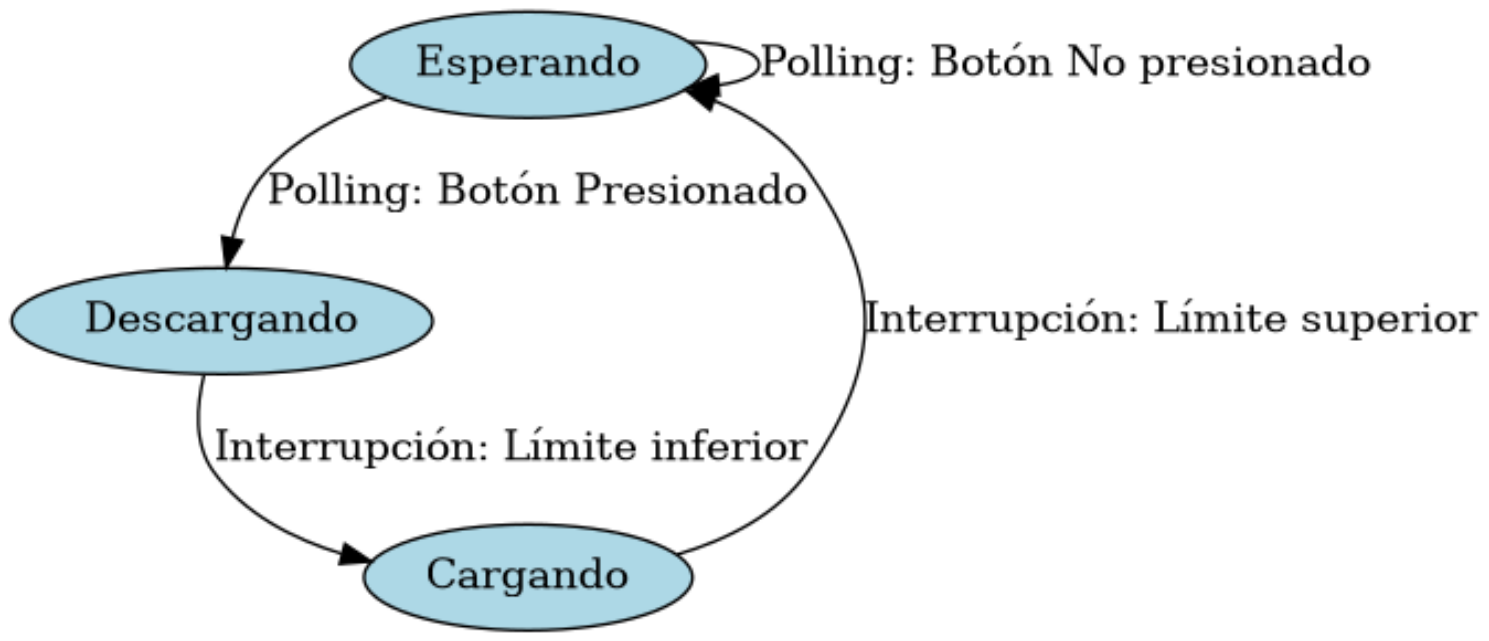
El dispositivo debe funcionar de la siguiente manera:

- Al presionarse el botón se debe abrir la exclusa para dejar circular el agua, y apagar el LED.
- Se debe mantener este estado hasta que el dispositivo sea interrumpido por pasarse del límite inferior de agua. En ese momento debe cerrarse la exclusa y abrirse la canilla.
- Tras ser interrumpido el dispositivo por haberse superado el límite superior de agua, debe cerrarse la canilla y prenderse el LED.

Se pide:

a) Describir los estados por los que pasa el sistema.
b) Mapear los registros de E/S a direcciones de E/S de la máquina ORGA1I.
c) Escribir el pseudocódigo de cómo opera la máquina ORGA1I.
d) Escribir en lenguaje ensamblador la rutina principal y la rutina de atención de interrupciones.

Solución 3.a - máquina de estados



:

Solución 3.a) y 3.b)

Estados:

- Esperando
- Descargando agua
- Cargando agua

Mapeo de registros:

- NIV_AGUA $\mapsto$ 0xFFF0
- LED_STATE $\mapsto$ 0xFFF1
- VAL_BOTON $\mapsto$ 0xFFF2
- SALIDA $\mapsto$ 0xFFF3
- CANILLA_ST $\mapsto$ 0xFFF4

Solución 3.c)

Pseudocódigo de la rutina principal

```
ESTADO = ESPERANDO;
while(true){
  if(ESTADO == ESPERANDO){
    SALIDA = 0xCE11;
    LED_STATE = 0xFFFF;
    CANILLA_ST = 0x00FF;
    if(VAL_BOTON == 0xB010){
      ESTADO = DESCARGANDO;
    }
  }
  else if(ESTADO == DESCARGANDO){
    SALIDA = 0xAB1E;
    LED_STATE = 0x0000;
    CANILLA_ST = 0x00FF;
  }
  else if(ESTADO == CARGANDO){
    SALIDA = 0xCE11;
    LED_STATE = 0x0000;
    CANILLA_ST = 0xADE7;
  }
}
```

Solución 3.c)

Pseudocódigo de la rutina de atención de interrupciones

```
if(NIV_AGUA == 0xA17A){
  ESTADO = ESPERANDO;
}
else if(NIV_AGUA == 0xBA1A){
  ESTADO = CARGANDO;
}
IRET();
```

Solución 3.d) - Rutina principal

```
; R1 --> ESTADO: 0x000 (esperando)
; 0x0001 (descargando), 0x0002(cargando)

            MOV R1, 0x0000          ;
inicio:     CMP R1, 0x0000          ; ¿esperando?
            JE esperando            ;
            CMP R1, 0x0001          ; ¿descargando?
            JE descargando          ;
            CMP R1, 0x0002          ; ¿cargando?
            JE cargando             ;
            JMP fin                 ;
esperando:  MOV [0xFFF3], 0xCE11    ; SALIDA
            MOV [0xFFF1], 0xFFFF    ; LED_STATE
            MOV [0xFFF4], 0x00FF    ; CANILLA_ST
            CMP [0xFFF2], 0xB010    ; ¿botón presionado?
            JE  cmb_estado          :
            JMP fin                 ;

descargando: MOV [0xFFF3], 0xAB1E   ; SALIDA
             MOV [0xFFF1], 0x0000   ; LED_STATE
             MOV [0xFFF4], 0x00FF   ; CANILLA_ST
             JMP fin                ;
cargando:    MOV [0xFFF3], 0xCE11   ; SALIDA
             MOV [0xFFF1], 0x0000   ; LED_STATE
             MOV [0xFFF4], 0xADE7   ; CANILLA_ST
             JMP fin                ;
cmb_estado:  ADD R1, 0x0001         ;
fin:         JMP inicio             ;
```

Solución 3.d) Rutina de atención de interrupciones

```
rut_ate_int: CMP [0xFFF0], 0xA17A ; NIV_AGUA
             JNE  otro            ;
             MOV R1, 0x0000       ; cambiamos el estado a esperando
otro:        CMP [0xFFF0], 0xBA1A ; NIV_AGUA
             JNE  fin_int         ;
             MOV R1, 0x0002       ; cambiamos el estado a cargando
fin_int:     IRET                 ;
```

Cierre

Cierre

Bibliografía:

- **Capítulo 7** *Computer Organization and Arquitecture* (Linda Null & Julia Lobur)

¿Cómo seguimos?

- Con esta clase ya tienen todo lo necesario para encarar el **Taller de Interrupciones**.

´