Entrada/Salida

Organización del Computador I

Facundo Linari → Mariana Milicich

Departamento de Computación - FCEyN
UBA

Primer Cuatrimestre 2019 - Turno Tarde

Hasta ahora vimos...

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

Hasta ahora vimos...

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

¿De qué sirve una computadora sin dispositivos de E/S?

Dispositivos de E/S

- **Entrada:**
- **Salida:**
- **Entrada/Salida:**

Dispositivos de E/S

- **Entrada:** Teclado, Mouse, Micrófono, etc.
- **Salida:**
- **Entrada/Salida:**

Dispositivos de E/S

- **Entrada:** Teclado, Mouse, Micrófono, etc.
- **Salida:** Monitor, Parlantes, Impresora, etc.
- **Entrada/Salida:**

Dispositivos de E/S

- **Entrada:** Teclado, Mouse, Micrófono, etc.
- **Salida:** Monitor, Parlantes, Impresora, etc.
- **Entrada/Salida:** Disco rígido, Módem, Placa de Red, etc.

Dispositivos de E/S

- **Entrada:** Teclado, Mouse, Micrófono, etc.
- **Salida:** Monitor, Parlantes, Impresora, etc.
- **Entrada/Salida:** Disco rígido, Módem, Placa de Red, etc.

¿Y cómo intercambio información con un dispositivo de E/S?

¿Cómo intercambio información con un dipositivo de E/S?

Cada dispositivo de E/S tiene sus propios registros, donde la CPU puede leer o escribir datos.

¿Cómo intercambio información con un dipositivo de E/S?

Cada dispositivo de E/S tiene sus propios registros, donde la CPU puede leer o escribir datos.

Tipos de registro:

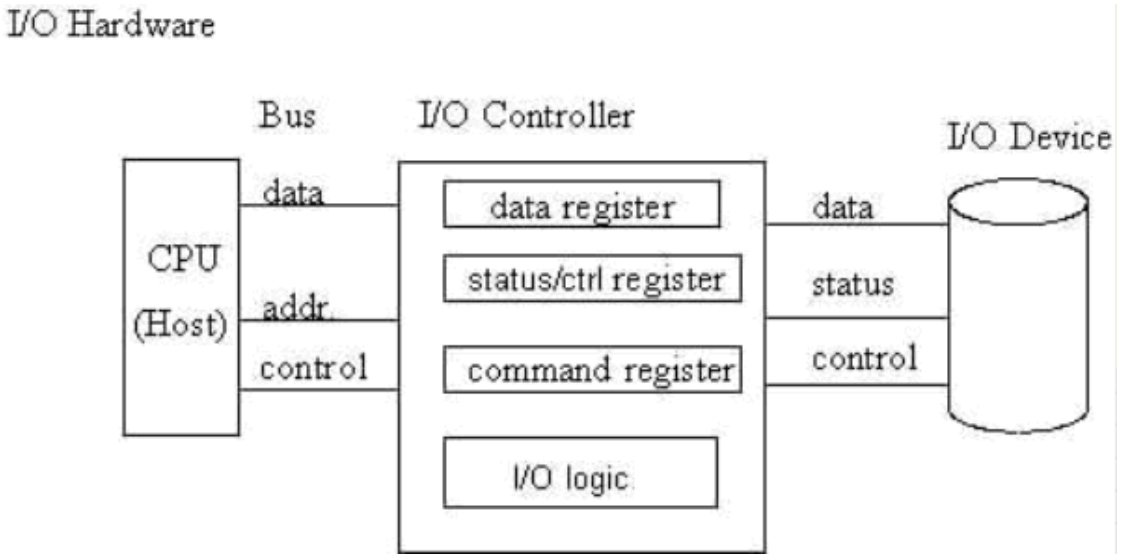
- Lectura.
- Escritura.
- Lectura/Escritura.

¿Cómo intercambio información con un dipositivo de E/S?

Cada dispositivo de E/S tiene sus propios registros, donde la CPU puede leer o escribir datos.

Tipos de registro:

- Lectura.
- Escritura.
- Lectura/Escritura.



Métodos de acceso a los registros

¿Con qué instrucciones accedo a estos registros?

Métodos de acceso a los registros

¿Con qué instrucciones accedo a estos registros?

Hay dos formas de referirse a los registros de un dispositivo de E/S:

Métodos de acceso a los registros

¿Con qué instrucciones accedo a estos registros?

Hay dos formas de referirse a los registros de un dispositivo de E/S:

- E/S independiente (instrucciones especiales: IN y OUT).
  Espacio de direcciones independiente

Métodos de acceso a los registros

¿Con qué instrucciones accedo a estos registros?

Hay dos formas de referirse a los registros de un dispositivo de E/S:

- 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

Métodos de acceso a los registros

¿Con qué instrucciones accedo a estos registros?

Hay dos formas de referirse a los registros de un dispositivo de E/S:

- 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

ORGA1 reserva las direcciones de memoria `0xFFF0` a `0xFFFF` para E/S.

Métodos de control de E/S

- **E/S por encuesta (Polling) o Programada**
- **E/S por interrupciones**
- **E/S por acceso directo a memoria (DMA)**

Métodos de control de E/S

- **E/S por encuesta (Polling) o Programada**
- **E/S por interrupciones**
- **E/S por acceso directo a memoria (DMA)**

¿Como funciona cada uno?

Métodos de control de E/S: Polling

- Cada dispositivo posee al menos un registro exclusivo que la CPU monitorea constantemente
- Cuando la CPU detecta un "data ready", actúa según lo que corresponda para ese registro
- La CPU está siempre chequeando

Métodos de control de E/S: Interrupciones

- La CPU no está constantemente chequeando
- El dispositivo interrumpe a la CPU: prende el flag de interrupción
- El S.O. debe interrumpir su tarea para atender al dispositivo. Para eso guarda la info de estado actual, carga la dirección de la rutina de atención de interrupción, ejecuta y al terminar restaura el estado

Métodos de control de E/S: DMA

- Los casos anteriores no resultan adecuados cuando se tienen dispositivos de E/S de alta velocidad o cuando hay que transferir grandes volúmenes de datos
- El controlador DMA es un dispositivo que controla una transferencia de datos entre un periférico y la memoria sin intervención de la CPU
- Es decir, el controlador DMA genera las señales necesarias para la transferencia de datos, especifica el tipo de operación (R/W), especifica la dirección de memoria de transferencia de datos, etc

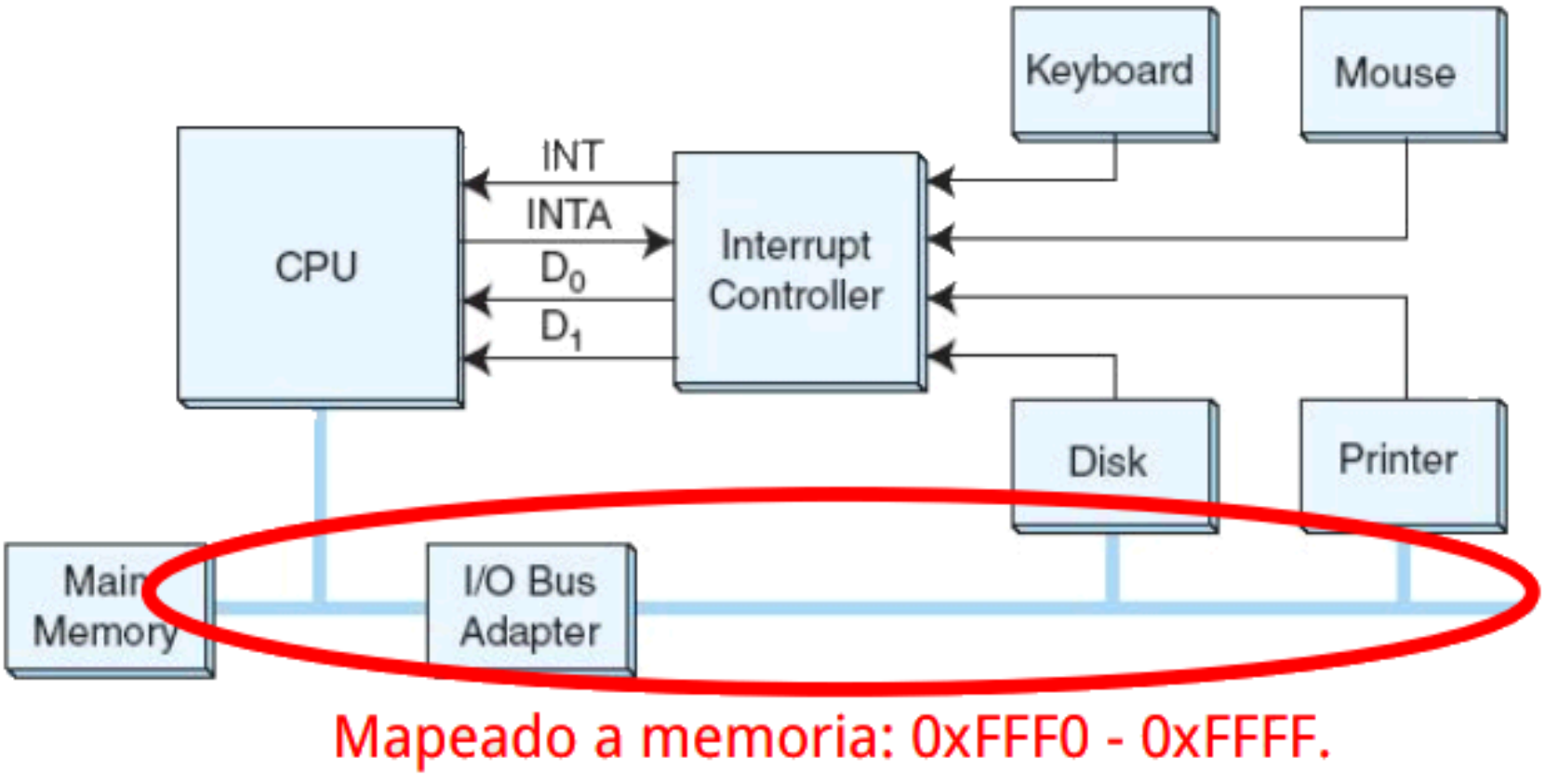
Ejercicio 1

Luego de muchos años de discusiones, la FIFA accedió a incorporar tecnología a los fallos de los árbitros en los partidos de fútbol. Gracias a ello nunca más se generará polémica a la hora de decidir si la pelota ingresó al arco o no.

Para lograr esto la FIFA, con su gran presupuesto, decidió comprar una máquina $\mathrm{ORGA1}$ junto con dispositivo de E/S llamado "LTA".

LTA cuenta con un registro de estado mapeado a la dirección de E/S `0xFFF0` de sólo lectura, en el cuál se refleja el porcentaje de la pelota ingresado al arco. Inicialmente, el registro se encuentra en el valor `0x0000` y aumenta a medida que la pelota va ingresando.
Escribir una rutina en ensamblador para satisfacer el sistema pedido, guardando la cantidad de goles en R0. Esta rutina se invoca al iniciar el partido.

Ejercicio 1 - ¿Qué sabemos de la máquina ORGA1?



Solución 1)

```
comienzaElPartido:   MOV R0, 0x0000
noEntro:             CMP [0xFFF0], 0x0064
                     JNE noEntro
                     ADD R0, 0x0001
sigueAdentro:        CMP [0xFFF0], 0x0000
                     JG sigueAdentro
                     JMP noEntro
```

El procesador ORGA1I

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

El procesador ORGA1I

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

- **Nuevas señales:**
  - Entrada: `INT` (Interrupción)
  - Salida: `INTA` (Interrupción reconocida)

El procesador ORGA1I

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

- **Nuevas señales:**
  - Entrada: `INT` (Interrupción)
  - Salida: `INTA` (Interrupción reconocida)
- **Nuevo flag:** `I`, que indica si el procesador puede ser interrumpido o no

El procesador ORGA1I

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

- **Nuevas señales:**
  - Entrada: `INT` (Interrupción)
  - Salida: `INTA` (Interrupción reconocida)
- **Nuevo flag:** `I`, que indica si el procesador puede ser interrumpido o no
- **Nuevo registro:** `PSW`, en donde se almacenan los flags

El procesador ORGA1I

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

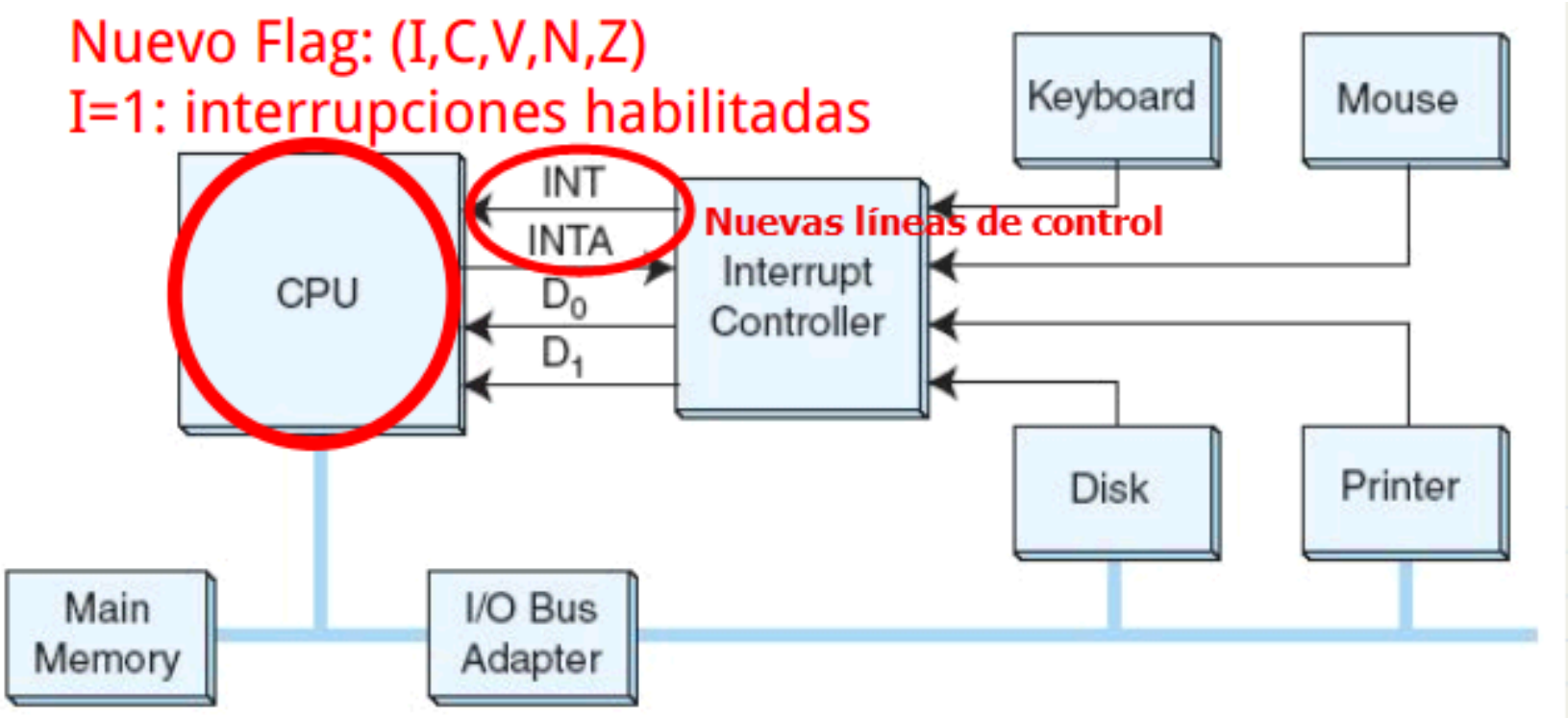
- **Nuevas señales:**
  - Entrada: `INT` (Interrupción)
  - Salida: `INTA` (Interrupción reconocida)
- **Nuevo flag:** `I`, que indica si el procesador puede ser interrumpido o no
- **Nuevo registro:** `PSW`, en donde se almacenan los flags
- **Nuevas instrucciones:**
  - `CLI` y `STI`
  - `IRET`
  - `PUSH Ri` y `POP Ri`

El procesador ORGA1I

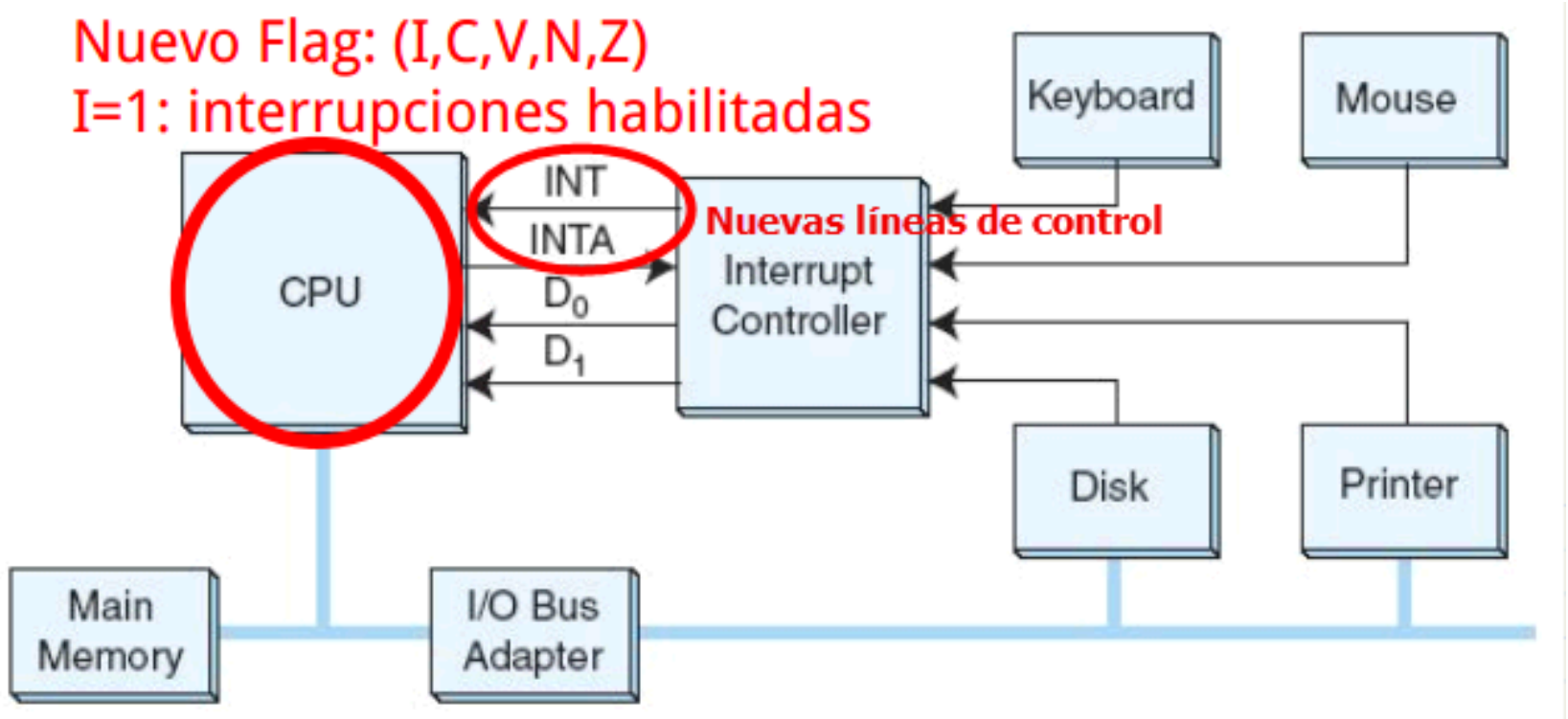
El procesador ORGA1I es un procesador ORGA1 que ha sido extendido con la capacidad de atender una única interrupción (enmascarable) de un dispositivo de E/S.

- **Nuevas señales:**
  - Entrada: `INT` (Interrupción)
  - Salida: `INTA` (Interrupción reconocida)
- **Nuevo flag:** `I`, que indica si el procesador puede ser interrumpido o no
- **Nuevo registro:** `PSW`, en donde se almacenan los flags
- **Nuevas instrucciones:**
  - `CLI` y `STI`
  - `IRET`
  - `PUSH Ri` y `POP Ri`
- **Nueva dirección reservada:** `0x0000`, donde se indica la dirección de la rutina de atención de la interrupción

El procesador ORGA1I



El procesador ORGA1i



Ver Teórica!!!

¿Qué pasa cuando interrumpen al CPU?

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

¿Qué pasa cuando interrumpen al CPU?

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

- Coloca [SP] = PSW y decrementa SP

¿Qué pasa cuando interrumpen al CPU?

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

- Coloca [SP] = PSW y decrementa SP
- Coloca [SP] = PC y decrementa SP

¿Qué pasa cuando interrumpen al CPU?

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

- Coloca [SP] = PSW y decrementa SP
- Coloca [SP] = PC y decrementa SP
- Coloca I=0 para evitar que el procesador vuelva a interrumpirse

¿Qué pasa cuando interrumpen al CPU?

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

- Coloca [SP] = PSW y decrementa SP
- Coloca [SP] = PC y decrementa SP
- Coloca I=0 para evitar que el procesador vuelva a interrumpirse
- Coloca PC = [0x0000]

¿Qué pasa cuando interrumpen al CPU?

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

- Coloca [SP] = PSW y decrementa SP
- Coloca [SP] = PC y decrementa SP
- Coloca I=0 para evitar que el procesador vuelva a interrumpirse
- Coloca PC = [0x0000]
- Activa la señal INTA para indicarle al dispositivo que atenderá su pedido

¿Qué pasa cuando interrumpen al CPU?

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

- Coloca [SP] = PSW y decrementa SP
- Coloca [SP] = PC y decrementa SP
- Coloca I=0 para evitar que el procesador vuelva a interrumpirse
- Coloca PC = [0x0000]
- Activa la señal INTA para indicarle al dispositivo que atenderá su pedido

Luego, comienza a ejecutarse la rutina de atención de la interrupción propiamente dicha

Ejercicio 2

Una computadora ORGA1I se está utilizando para monitorear el estado de una montaña rusa. Esta computadora se encarga de verificar ciertos parámetros de la montaña rusa y actuar de acuerdo a su estado. Para ello, cuenta con dos dispositivos de E/S que actúan como sensores, uno que actúa como alarma y otro que efectúa una parada de emergencia. Cada sensor posee un registro de E/S de sólo lectura que reportan la siguiente información:

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

Ejercicio 2 (continuación)

Las etiquetas MAX_SPEED y MIN_BRAKES son constantes de 16 bits.

El dispositivo de alarma posee un registro de E/S de lectura/escritura (ALARMA) que permite activar las alarmas correspondientes.

El bit menos significativo representa la alarma de velocidad. Esta le indica al operador que el carrito está yendo muy rápido. El segundo bit representa el estado de los frenos comunes. Este indica que hay un problema con estos frenos y que se van a activar los frenos de emergencia. Si el bit está en 1, la alarma está encendida. Las alarmas se apagan de manera externa al sistema.

El dispositivo de frenos de emergencia posee un registro de E/S de escritura (FRENOS_EM) que activa los frenos de emergencia en caso que se detecte un problema con los frenos comunes. Para ello hay que setear todos los bits en 1.

Ejercicio 2

a) Mapear los registros de E/S a direcciones de E/S de ORGA1.

b) Realizar el código para sensar y activar las alarmas correspondientes.

c) Suponiendo que el ciclo de instrucción de cada instrucción del programa tarda 3 ms y los valores máximos nunca se alcanzan ¿Cuál es la frecuencia (en Hz) de muestreo (lectura) de los sensores? ¿Y si todos los sensores sobrepasan los máximos?

Solución 2.a)

Mapeo de registros.

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

Solución 2.b)

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

Solución 2.c)

Si no se alcanzan los valores máximos, cada iteración ejecuta 4 instrucciones. Por lo tanto, podemos concluir que cada iteración tarda $4 * 3$ ms. Como podemos realizar una única lectura por iteración, se lee cada señal cada $4 * 3$ ms. En conclusión (y teniendo en cuenta que 1000ms= 1s), la frecuencia de muestreo es $\frac{1000}{4*3} = 83,33$ Hz.

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

Ejercicio 3

El dueño de la montaña rusa invirtió algo de dinero y compró un nuevo sensor para los frenos. Este nuevo sensor solicita una interrupción si se detecta un inconveniente con los frenos comunes.

a) Modificar el programa presentado para aprovechar esta característica de modo que la frecuencia de muestreo sea mayor.
b) Calcular la nueva frecuencia de muestreo para el sensor de velocidad.
c) Escribir la rutina de atención de la interrupción del sensor de frenos.
d) ¿Cómo quedaría la memoria si se pide además, que la rutina principal se cargue a partir de la posición 0xAAAA y la rutina de atención de interrupciones 20 posiciones antes?

Solución 3)

Modificación

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

Solución 3)

Modificación

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

Muestreo

Si no se alcanza el valor máximo, cada iteración ejecuta 2 instrucciones. Dando una frecuencia de muestreo de $\frac{1000}{2*3} = 166,66$ Hz. Si, en cambio, se sobrepasa el valor máximo, cada iteración ejecuta 4 instrucciones, dando una frecuencia de $\frac{1000}{4*3} = 83,33$ Hz.

Solución 3)

Modificación

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

Muestreo

Si no se alcanza el valor máximo, cada iteración ejecuta 2 instrucciones. Dando una frecuencia de muestreo de $\frac{1000}{2*3} = 166,66$ Hz. Si, en cambio, se sobrepasa el valor máximo, cada iteración ejecuta 4 instrucciones, dando una frecuencia de $\frac{1000}{4*3} = 83,33$ Hz.

Rutina de atención de la interrupción

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

Solución 3.d)

- A partir de la posición de memoria 0xAAAA se carga la rutina principal.

Solución 3.d)

- A partir de la posición de memoria 0xAAAA se carga la rutina principal.
- A partir de la posición de memoria 0xAA96 se carga la rutina de atención de interrupciones.

Solución 3.d)

- A partir de la posición de memoria 0xAAAA se carga la rutina principal.
- A partir de la posición de memoria 0xAA96 se carga la rutina de atención de interrupciones.
- En la posición 0x0000 cargamos 0xAA96, que es la posición de memoria donde comienza la rutina de atención de interrupciones.

Ejercicio 4

Se está desarrollando un novedoso dispositivo. Su funcionalidad es almacenar agua en un tanque y dejar pasar la misma por una exclusa, cuando el operario de la máquina lo disponga. Además, posee un sistema de llenado automático y una luz para avisar al operario que el tanque está lleno. Se desea controlar este dispositivo con una ORGA1I. Para lograr esto el sistema posee:

- un botón que puede ser apretado por el usuario. Su estado se ve reflejado en el registro `VAL_BOTON`: si está presionado vale `0xB010`, caso contrario `0x0000`.
- un `LED` que se debe encender cuando el tanque está lleno para indicar que el sistema está listo para volver a usarse. Para encenderlo hay que escribir `0xFFFF` en el registro `LED_STATE`, y para apagarlo hay que escribir `0x0000` en dicho registro.
- una `EXCLUSA` que permite dejar pasar el agua. Para que la misma esté abierta 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 se pasa de los límites permitidos. Al producir la interrupción, además, escribe `0xA17A` en el registro `NIV_AGUA` si la interrupción se produjo porque el agua superó en nivel alto, y escribe `0xBA1A` si fue por el bajo.
- una `CANILLA` que deja entrar agua al tanque si escribimos `0xADE7` en el registro `CANILLA_ST`, para cerrar la canilla hay que escribir `0x00FF` en el mismo registro.

Las funcionalidades del dispositivo deben ser las siguientes: al apretarse el botón, se debe abrir la exclusa que deja circular el agua y apagar el LED. El sistema mantendrá este estado hasta que se produzca una interrupción por el nivel bajo del agua. En ese momento se debe cerrar la exclusa, y abrir la canilla. Cuando se vuelva a producir una interrupción por el nivel alto de agua, se debe cerrar la canilla y prender el led para avisar al usuario que el sistema esta listo para usarse nuevamente.
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 ORGA1I.
c) Escribir el pseudocodigo de la máquina ORGA1I.
d) Escribir en leguaje ensamblador la rutina principal y la rutina de atención de interrupciones.

Solución 4.a) y 4.b)

Estados: Esperando, Descargando agua, Cargando agua

Solución 4.a) y 4.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 4.c)

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 4.c)

Rutina Atención Interrupciones

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

Solución 4.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  ; Como estamos esperando, chequeamos si se presionó el botón
            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 4.d)

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

bibliografia

Computer Organization and Arquitecture, Linda Null and Julia Lobur

Capítulo 7

¿Cómo seguimos?

- Ya pueden hacer (toda) la práctica 5: Entrada/Salida

¿Preguntas?