Entrada/Salida

Orga 1

Departamento de Computación
Facultad de Cs Exactas y Naturales
Universidad de Buenos Aires

1 de marzo de 2018

Hasta ahora vimos...

- Representación de la información
- Lógica digital
- Formato de instrucción
- Microprogramación y algo de assembler de $\mathrm{ORGA1}$
- Cache

Nos falta un mecanismo para comunicarnos con el mundo exterior

Dispositivos de E/S

- **Entrada:** teclado, mouse, micrófono, etc.
- **Salida:** monitor, parlantes, impresora, etc.
- **Entrada/Salida:** disco rígido, lector de tarjetas, placa de red, etc.

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

¿Qué es un dispositivo de E/S para la CPU?

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

Tipos de registro:

- Lectura
- Escritura
- Lectura/Escritura

¿Qué datos se pueden escribir o leer?

Métodos de acceso a los 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.

Esquemas de E/S

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

¿Cómo funciona cada uno?

Ejercicio 1

En la lejana república de Xor hay demasiadas cabinas telefónicas que ocupan toda la vereda y no dejan caminar a la gente. El gobierno querría sacar algunos teléfonos del medio del paso, por lo que decidió hacer una campaña para averiguar cuáles son los teléfonos que menos se usan.

Para lograr esto el gobierno decidió equipar a las cabinas telefónicas con una máquina ORGA1 conectada a un dispositivo de E/S que avisa si hay una persona adentro de la cabina o no.

Este sensor cuenta con un registro de estado mapeado a la dirección de E/S `0xFFF0` de sólo lectura, en el cual se refleja el porcentaje de persona adentro de la cabina. Inicialmente, el registro se encuentra en el valor `0x0000` y aumenta a medida que el usuario atraviesa la puerta de la cabina.

Escribir una rutina en ensamblador que use estas herramientas para contar (en `R0`) la cantidad de gente que usa una cabina dada.

Solución

```
comienzaElDia:    MOV R0, 0x0000
nadieAdentro:     CMP [0xFFF0], 0x0064
                  JNE nadieAdentro
                  ADD R0, 0x0001
alguienAdentro:   CMP [0xFFF0], 0x0000
                  JG alguienAdentro
                  JMP nadieAdentro
```

El procesador ORGA1i

El procesador ORGA1i es un procesador ORGA1 que ha sido extendido con la capacidad para atender la interrupción (enmascarable) de un único dispositivo de E/S.

- Nuevas señales:
  - Entrada: `INTR` (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

¿Qué pasa cuando interrumpen al CPU?

En el caso de un procesador 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 $\mathrm{ORGA1i}$ 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 reporta 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 t s 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 Ejercicio 2.a)

Mapeo de registros.

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

Solución Ejercicio 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 Ejercicio 2.c)

Si no se alcanzan los valores máximos, cada iteración ejecuta 4 instrucciones. Por lo tanto, podemos concluir que cada iteración tarda $4 * t$ s. Como podemos realizar una única lectura por iteración, se lee cada señal cada $4 * t$ s. En conclusión, la frecuencia de muestreo es $\frac{1}{4*t}$ Hz.

En cambio, si se sobrepasan los valores máximos, cada iteración ejecuta 8 instrucciones, tardando $8 * t$ s. Cada señal se lee cada $8 * t$ s, dando una frecuencia de $\frac{1}{8*t}$ 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.

I. Modificar el programa presentado para aprovechar esta característica de modo que la frecuencia de muestreo sea mayor.
II. Calcular la nueva frecuencia de muestreo para el sensor de velocidad.
III. Escribir la rutina de atención de la interrupción del sensor de frenos.

Solución ejercicio 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 $\frac{1}{2*t}$ Hz. Si, en cambio, se sobrepasa el valor máximo, cada iteración ejecuta 4 instrucciones, dando una frecuencia de $\frac{1}{4*t}$ Hz.

Rutina de atención de la interrupción

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

El procesador 8086 (simplificado)

El procesador 8086 es similar a ORGA1i , pero tiene algunas diferencias significativas.

- Procesador de 16 bits con direccionamiento a byte
- Registros:
  - 8 registros de propósito general: `AX`, `BX`, `CX`, `DX`, `BP`, `SP`, `DI`, `SI`
  - `AL`, `AH`, `BL`, `BH`, `CL`, `CH` y `DL` y `DH` para acceder a la parte baja y alta respectivamente de `AX`, `BX`, `CX` y `DX`.
- Tiene las mismas instrucciones que ORGA1i , y algunas más
- Modos de direccionamiento: solo directo, no tiene instrucciones memoria a memoria
- **Los registros de E/S se encuentran en un espacio de direcciones diferente**
- Instrucciones para acceder a los registros de E/S:
  - `IN Reg, RegES`
  - `OUT RegES, Reg`

8086 + PIC 8259

El procesador 8086 provee soporte para interrupciones por medio del PIC 8259.

- 8 interrupciones enmascarables, de la `IR0` a la `IR7`
- 2 registros de E/S:
  - `IRR` (Interrupt Request Register): el i-ésimo bit indica si la i-ésima línea de interrupción es activada
  - `IMR` (Interrupt Mask Register): 8 bits, el i-ésimo bit indica si la i-ésima interrupción debe ser atendida o no
- Interrupciones en 8086:
  - Activa la señal `INTA`
  - El PIC coloca en el bus en número de interrupción correspondiente
  - El CPU indexa el vector de interrupciones (a partir de `0x0000`) con el número de interrupción
  - Pushea `PSW` y `PC`
  - Deshabilita interrupciones
  - Modifica `PC` con la dirección de la rutina de atención

Ejercicio 4

I. Reescribir la rutina de sensado y activación de alarmas del ejercicio anterior asumiendo que cambia la arquitectura de la computadora de la montaña rusa por un 8086. Asumir que los registros de E/S están mapeados a los mismos valores que antes.

II. Reescribir la rutina de antención de interrupciones para dicha arquitectura.

Solución ejercicio 4

Sensado

```
sensaVel:   IN AX, 0xFFF0
            CMP AX, MAX_SPEED ;alcanzó velocidad máxima?
            JL sensaVel
            IN AX, 0xFFF2
            OR AX, 0x0001
            OUT 0xFFF2, AX
            JMP sensaVel
```

Rutina de atención de la interrupción

```
rut_at_int:   PUSH AX
              MOV AX, 0xFFFF
              OUT 0xFFF3, AX
              IN AX, 0xFFF2
              OR AX, 0x0002
              OUT 0xFFF2, AX
              POP AX
              IRET
```

Ejercicio 5

En una computadora ORGA1i se ha conectado un controlador DMA. El acceso a cada uno de los registros del controlador está mapeado a memoria del siguiente modo:

DEVICE	DEVICE_ADDRESS	MEM_ADDRESS	SIZE	STATUS
0xFFF0	0xFFF1	0xFFF2	0xFFF3	0xFFF4

El identificador del disco rígido es 0x354A. Se quiere transferir desde la posición 0x0045 del disco hasta la 0x013A. Estos datos se deben guardar a partir de la posición 0xA142 de la memoria principal. El bit menos significativo del registro STATUS contiene un 1 en caso de escritura, y un 0 en caso contrario. Para indicarle al DMA que ya se cargaron todos los datos y que puede iniciar la transmisión, se setea en 1 el bit más significativo del registro STATUS. El resto de los bits del registro STATUS se encuentran reservados. El DMA contiene un registro interno TEMP en el que puede almacenar datos localmente.

Ejercicio 5 (continuación)

a) Escribir un programa en *assembler* que transfiera los datos indicados en el enunciado utilizando el DMA.

b) Describir con un pseudo-código el comportamiento del dispositivo DMA.

Solución 5.a)

```
Inicio:  MOV [0xFFF0], 0x354A   ; [DEVICE]
         MOV [0xFFF1], 0x0045   ; [DEVICE_ADDRESS]
         MOV [0xFFF2], 0xA142   ; [MEM_ADDRESS]
         MOV [0xFFF3], 0x00F5   ; [SIZE]
         AND [0xFFF4], 0xFFFE   ; [STATUS]
         OR [0xFFF4], 0x8000    ;
```

Notar que es importante que el bit más significativo de *STATUS* sea el último en escribirse ya que le indica al DMAC que comience la transferencia.

Solución 5.b)

```
while(true){
    while (STATUS[15] != 1) { //esperar  }
    if (STATUS[0] == 0) { //Disco a memoria
        while(SIZE > 0) {
            TEMP <- LeerIO(DEVICE, DEVICE_ADDR)
            EscribirMEM(TEMP, MEM_ADDR)
            MEM_ADDR++
            DEVICE_ADDR++
            SIZE--
        }
    }
    else{
        //Tarea (memoria a disco)
    }
    STATUS[15] <- 0
    INT()
}
```

¿Preguntas?