(a) Función de los registros del PIC

El **Controlador Programable de Interrupciones (PIC)** es un dispositivo encargado de gestionar las interrupciones en los sistemas basados en la arquitectura **x86**. Se comunica con la CPU y maneja las interrupciones externas a través de diversos registros.

Registro	Función	Dirección
IRR (Interrupt Request Register)	Almacena las interrupciones pendientes que han sido detectadas.	`20H`
ISR (In-Service Register)	Indica qué interrupciones están siendo atendidas en ese momento.	`20H`
IMR (Interrupt Mask Register)	Habilita o bloquea ciertas interrupciones. Un bit en `1` bloquea la interrupción correspondiente.	`21H`
INT0 - INT7	Son las líneas de interrupción conectadas a los dispositivos externos. Cada línea maneja un tipo de interrupción diferente (teclado, reloj, etc.).	`20H` - `27H`
EOI (End of Interrupt)	Indica al PIC que la interrupción ha sido atendida y se puede procesar la siguiente. Se envía al **registro de comando** del PIC.	`20H`

📌 **¿Cómo se usa cada registro en el código?**

- `IMR (PIC+1 = 21H)` : Se configura con `0FEH` para habilitar solo la línea `INT0` correspondiente a la interrupción de la tecla **F10**.
- `INT0 (PIC+4 = 24H)` : Se configura con `N_F10` para manejar la interrupción.
- `EOI (20H)` : Se usa en `RUT_F10` para indicar que la interrupción ya fue atendida.

(b) Registros programables y la instrucción OUT

Los registros **programables** son aquellos que pueden ser configurados mediante instrucciones **OUT** para controlar el comportamiento del PIC. En este caso, los registros programables son:

1. **IMR (Interrupt Mask Register -** `21H`**)**: Se usa para habilitar o deshabilitar interrupciones.
2. **Comando de inicialización (** `20H` **)**: Se usa para enviar configuraciones especiales.
3. **Registro EOI (** `20H` **)**: Se usa para finalizar la interrupción actual.

¿Cómo trabaja la instrucción `OUT`?

La instrucción `OUT` se usa para enviar datos a un puerto de E/S (Entrada/Salida). Su sintaxis es:

assembly Copiar Editar

```
OUT dirección, AL
```

Donde:

- `dirección` es la dirección del puerto donde se escribirá el valor.
- `AL` es el registro que contiene el valor a escribir.

Ejemplos en el código:

```assembly
MOV AL, 0FEH
OUT PIC+1, AL ; Configura el IMR del PIC

MOV AL, N_F10
OUT PIC+4, AL ; Configura la línea de interrupción INT0

MOV AL, EOI
OUT EOI, AL   ; Indica que la interrupción terminó
```

En cada caso, la instrucción **OUT** transfiere el valor de `AL` al puerto especificado.

(c) Instrucciones CLI y STI

Estas instrucciones controlan el estado de las interrupciones en el procesador.

Instrucción	Función
`CLI` **(Clear Interrupt Flag)**	Deshabilita todas las interrupciones en la CPU (bit IF en 0).
`STI` **(Set Interrupt Flag)**	Habilita las interrupciones en la CPU (bit IF en 1).

¿Cómo se usan en el código?

```assembly
CLI  ; Deshabilita interrupciones mientras se configura el PIC
...
STI  ; Habilita interrupciones después de la configuración
```

- `CLI` evita que se produzcan interrupciones mientras se configura el PIC.
- `STI` permite que las interrupciones se procesen nuevamente después de la configuración.

Resumen del flujo del programa

1. **Se configura el PIC:**
   - Se habilita la interrupción de la tecla **F10** (`N_F10`).
   - Se enmascaran las demás interrupciones (`IMR`).
2. **Se entra en un lazo infinito** (`LAZO`).
3. **Cuando se presiona F10**, se ejecuta la rutina de interrupción `RUT_F10`:
   - Se incrementa `DX` (contador de presiones de F10).
   - Se envía `EOI` al PIC para permitir nuevas interrupciones.
4. **El programa sigue corriendo indefinidamente**.

Este código es una implementación básica de manejo de interrupciones por hardware utilizando el PIC.