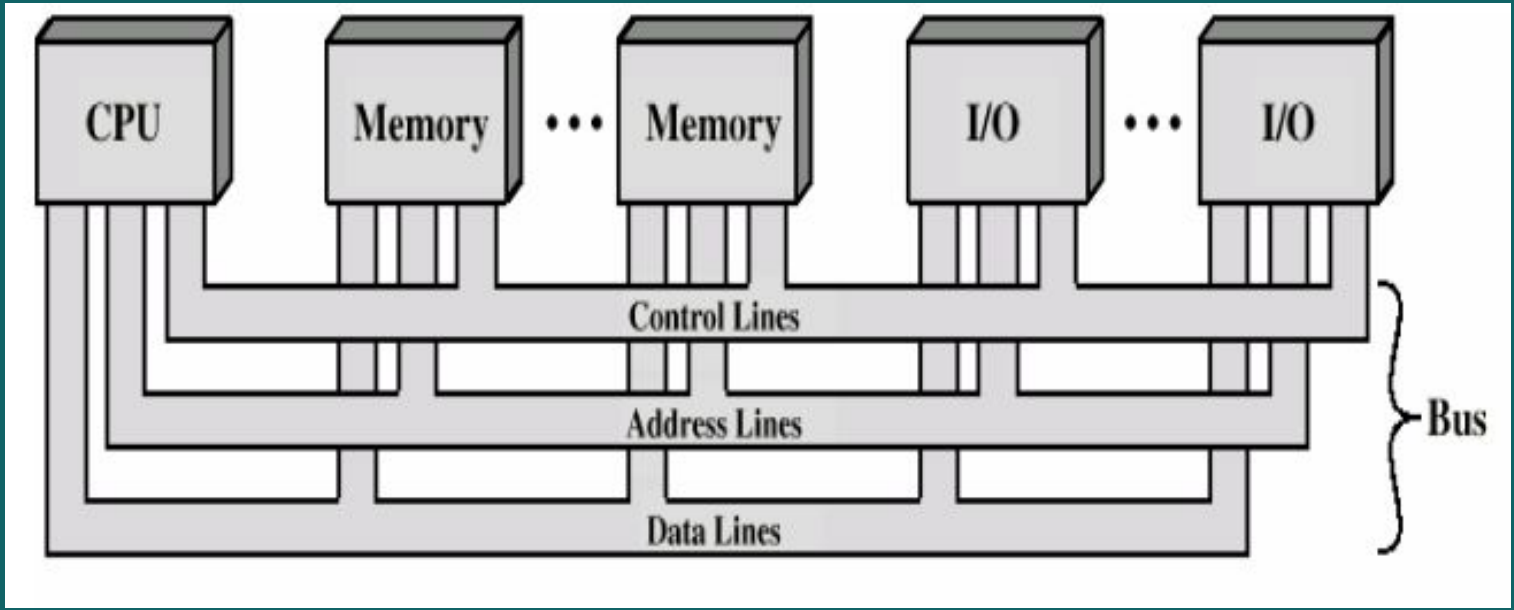
Interrupciones y E/S

Arquitectura de Computadoras - Práctico 9

Contextualizando



Aspectos relevante

- E/S Mapeada en memoria o aislada.
- Tres formas de interactuar con E/S
  - E/S Programada
  - Interrupciones
  - DMA

Aspectos programación

- Funciones:
  - dato = in(dir)
  - out(dir, dato)
  - enable()
  - disable()
- Registros de E/S:
  - Tipo (lectura, escritura o ambos)
  - Tamaño (byte o palabra)

Interrupción

Mecanismo que provoca la alteración del orden lógico de ejecución de instrucciones como respuesta a un evento externo, generado por el hardware de E/S, en forma asincrónica al programa que esté siendo ejecutado y fuera de su control.

La próxima instrucción a ser ejecutada NO es la que corresponde a la secuencia lógica de instrucciones del programa, sino que se ejecuta la primer instrucción de la rutina de servicio de la interrupción.

Programación

¿Qué se programa?

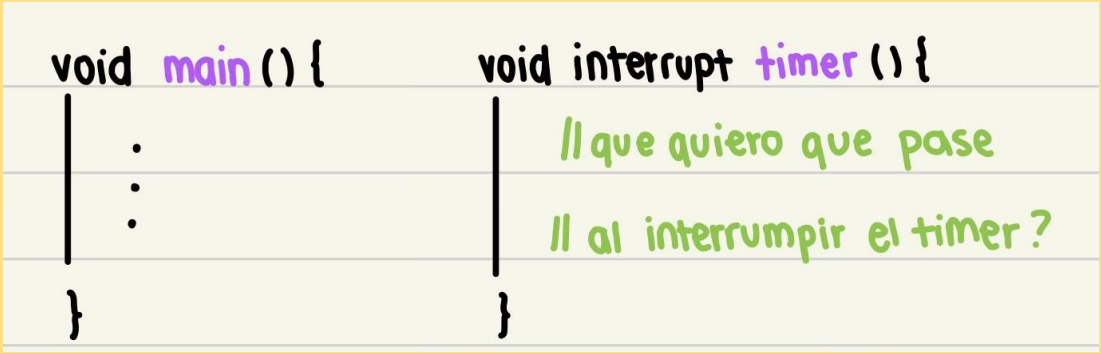
Un sistema que contiene rutinas de interrupción

¿Cómo?

Por lo general, se tiene un programa principal (void main( )) y una o más rutinas de interrupción asociadas a los eventos de dispositivos E/S que deben ser considerados por el sistema. Cuando se da una interrupción, se ejecute el código contenido en su rutina de interrupción asociada

Programación

Por ejemplo: tengo un timer que ejecuta una rutina de interrupción cada X tiempo, entonces tengo una rutina void interrupt timer( )



Máquina dedicada

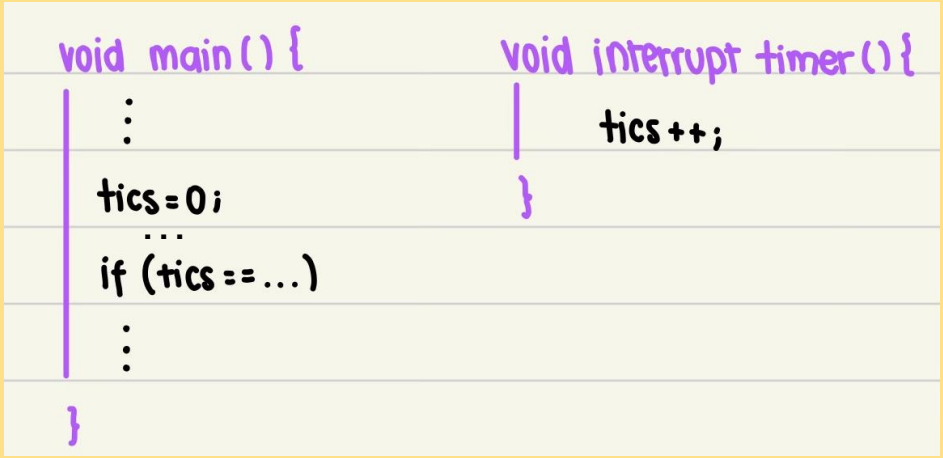
Máquina dedicada: la máquina está completamente dedicada a una función específica y todos los programas y rutinas que en ella se ejecutan están desarrollados por el mismo equipo de programadores o por equipos fuertemente coordinados.

Máquina NO dedicada: donde se ejecuta la rutina existen múltiples programas, con distintos propósitos, los cuales han sido programados por distintos equipos, sin coordinación.

Lógica del sistema

La lógica del sistema puede estar en diferentes lugares:

- Opción 1 (solo para màquinas dedicadas): Toda la lógica en el programa principal y las rutinas de interrupción solamente registra que se ejecuta modificando una variable global.



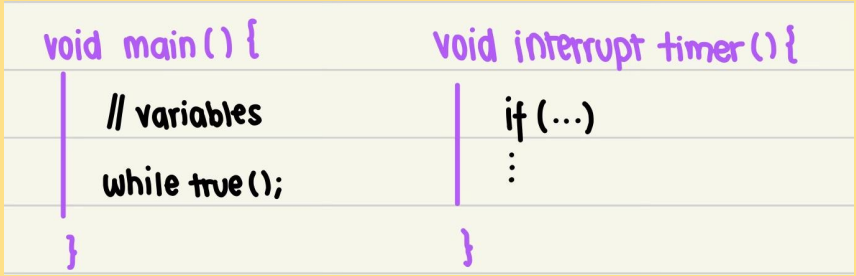
Lógica del sistema

- Opción 2: Toda la lógica se implementa en las rutinas de interrupción y el programa principal varía según si la máquina es dedicada o no.

- Opción 3: soluciones hìbridas.

Lógica del sistema

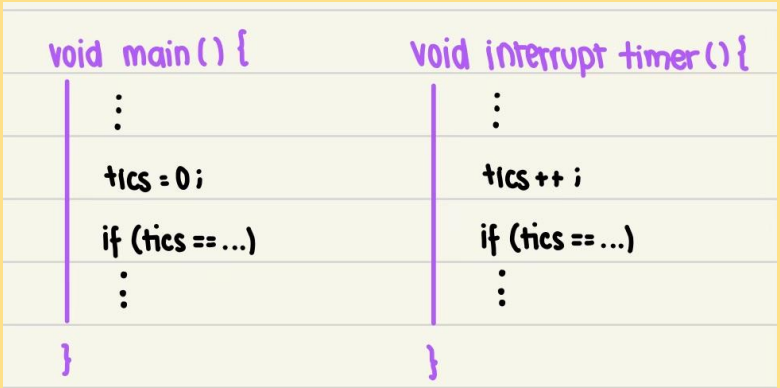
- Toda la lógica se implementa en las rutinas de interrupción. El programa principal:
  - En una máquina dedicada debe incluir un loop infinito (la máquina se dedica solo a esa tarea)



  - En una máquina no dedicada instala e inicializa las rutinas de interrupción y termina (la lógica está en las rutinas)

Lógica del sistema

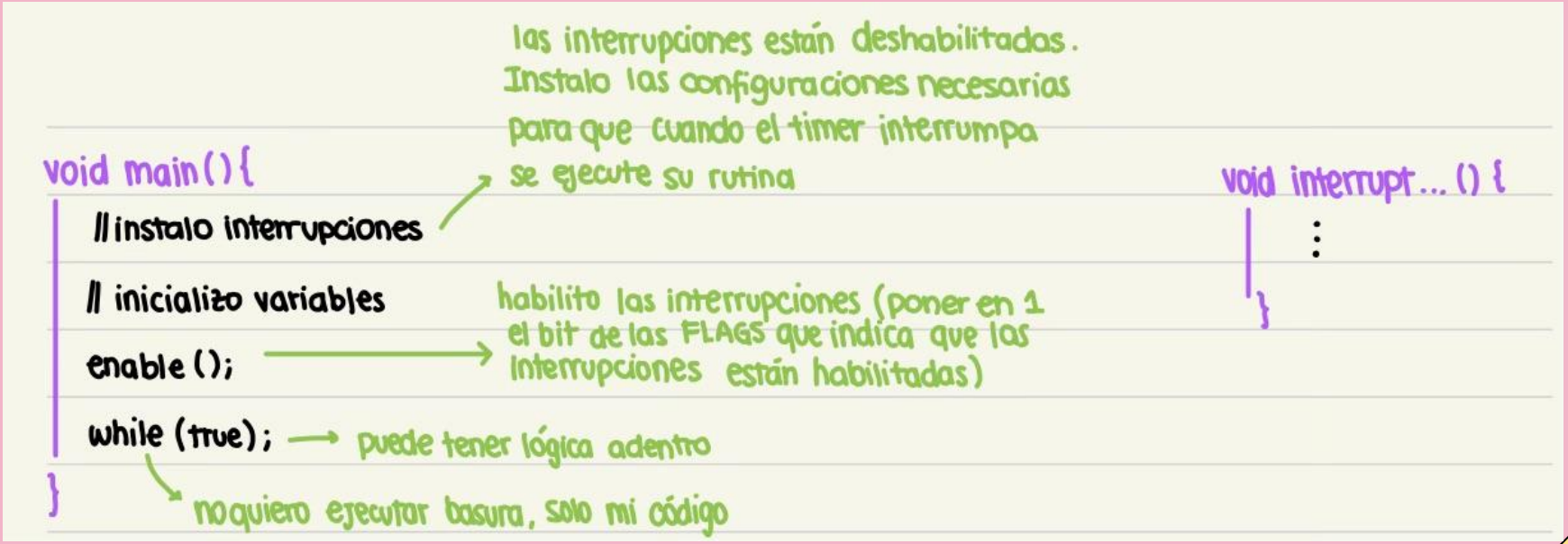
- La lógica se implementa parte en el programa principal y parte en las rutinas de interrupción.



NO HAY una regla que indique cuál es mejor.

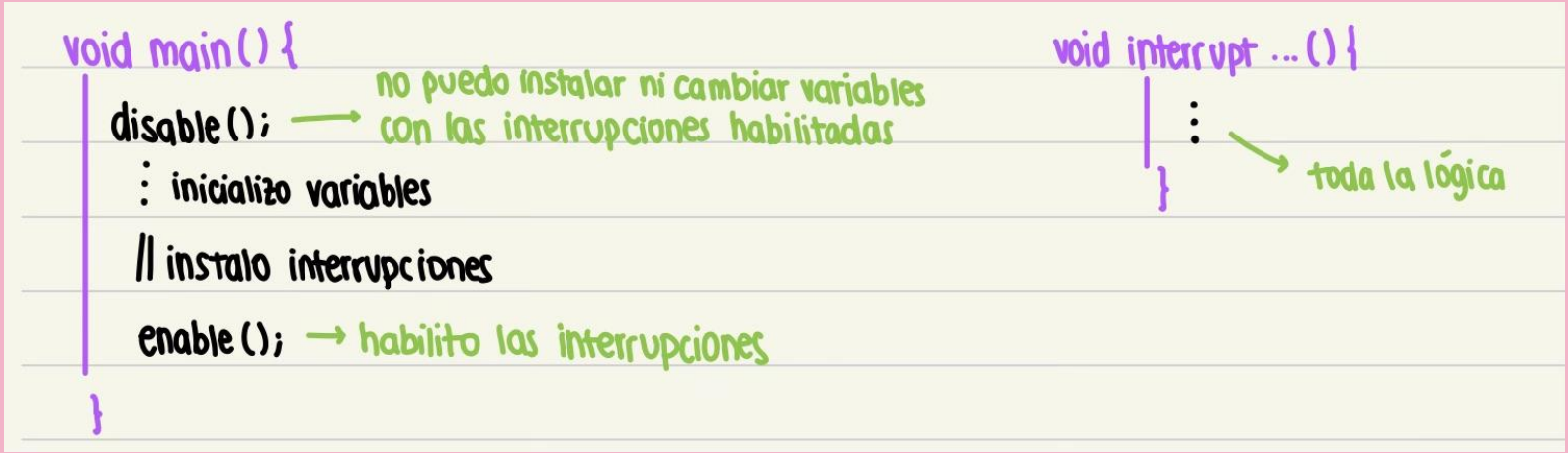
Según tipo de máquina:

- Máquina dedicada: la lógica puede estar en el main, en las interrupciones o en ambas, ya que la máquina se dedica solo a una tarea que ejecuta con el while (true).



Según tipo de máquina:

- Máquina NO dedicada: la lógica tiene que estar en las interrupciones, no se incluye while (true) porque se capturaría la CPU, lo cual no corresponde a ambientes no dedicados.



Variables y tipos

- Muchas veces es útil hacer una máquina de estados:

```
# define estado1 = 0;
# define estado2 = 1;
      ⋮
```

y hacer un switch :

```
switch (variable) {
    case estado1:
        ...
    case estado2:
        ...
    default:
        ...
}
```

voy cambiando de estado según mi máquina

- También podemos usar estructuras o arreglos para modelar distintas cosas:

```
struct {
    int cedula;
    int edad;
} personas [50];
```

```
int velocidad [4];
int maximo [] = {0x02, 0x08, 0x20}
```

Variables y tipos

- Operadores BIT a BIT en C:

  - `&` : and bit a bit (0110 & 0100 = 0100)
  - `|` : or
  - `>>` : shift right (0111 >> 1 = 0011)
  - `<<` : shift left
  - `^` : xor
  - `~` : not (complemento a 1)

- Tiempo:

→ Si tengo un timer de 10Hz: el timer interrumpe 10 veces por segundo

Ejemplo: quiero contar 2 min

```
void interrupt timer () {
    tics ++;
}
```

2 min ~ $2 * \frac{60}{\text{seg}} * \frac{10}{\text{timer}}$ tics ⇒ if (tics = 1200)
⋮

→ 10 KHz ~ 100.000 veces por segundo

→ $1ms = 0{,}001s = 10^{-3}s$

$1\mu s = 0{,}000001s = 10^{-6}s$

$1ns = 10^{-9}s$

Puertos de E/S

¿Cómo leemos y escribimos puertos de E/S?

→ **solo lectura :**

- leemos con `in()`, por ejemplo tengo un puerto de E/S de 16 bits de solo lectura en la dirección DECIMETROS que contiene... entonces hago `int valor = in(DECIMETROS);`
- a veces, podemos tener más de un valor en el puerto, por ejemplo : los dos bits menos significativos contienen ..., los siguientes cuatro ... Para obtenerlos usamos máscaras, por ejemplo, para obtener los dos menos significativos hago `valor & 0x03`

Puertos de E/S

→ solo escritura:

- escribimos con out(), por ejemplo tengo un puerto de E/s de 16 bits de solo escritura en la dirección DISTANCIA donde se debe escribir ... entonces hago out(DISTANCIA, valor);
  - valor: lo que quiero escribir

Puertos de E/S

→ **lectura y escritura** : tengo que conservar los bits que no se de que son

ejemplo: válvula dispensadora, puerto E/S byte lectura y escritura en la dirección DISPENSADOR

El bit 1 controla la válvula de café, el 2 leche (1 abierta, 0 cerrada)

```
out (DISPENSADOR, in (DISPENSADOR) & 0x FD)  // cierro válvula de café
                                    ↓
                                  1111 1101
```

Con esto dejo los otros valores incambiados menos el que yo quería modificar

Ejercicio 1 y 2

¿Cuál es el desafío en estos ejercicios?

E/S programada

Ejercicio 3 y 4

¿Cuál es el desafío en estos ejercicios?

Ejercicios atípicos

Entender algunos aspectos a resolver a bajo nivel

Ejercicio 5 a 10

¿Cuál es el desafío en estos ejercicios?

Desarrollos de soluciones basadas en interrupciones.