Laboratorio 1:

# Acceso elemental a dispositivos (i)

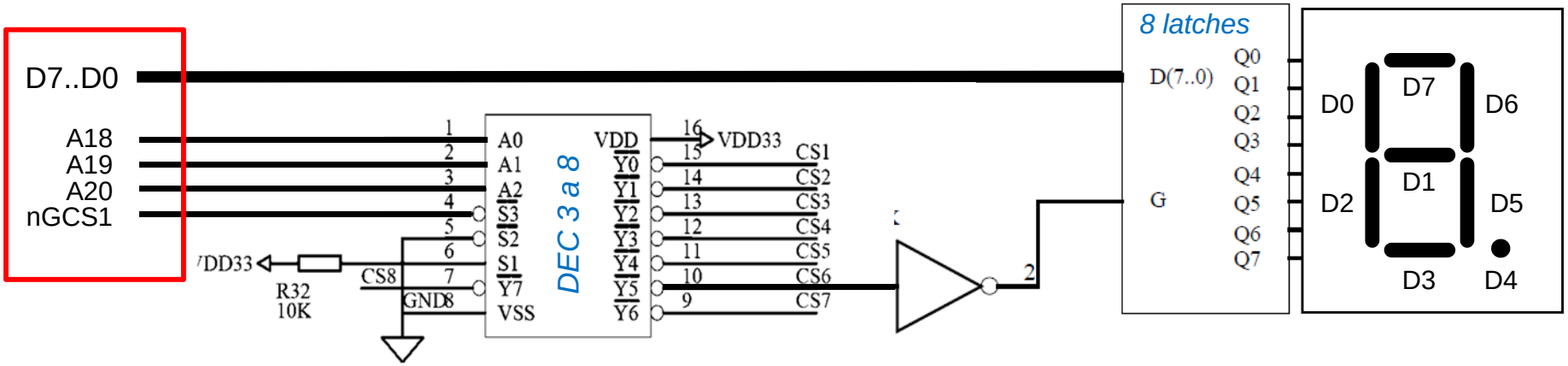
dispositivos externos mapeados en memoria

Programación de sistemas y dispositivos

**José Manuel Mendías Cuadros**
*Dpto. Arquitectura de Computadores y Automática*
*Universidad Complutense de Madrid*

# Conexión del display

- Al bus de memoria del SoC hay conectado un display 7-segmentos:




- El array de latches que estimulan los segmentos (lógica inversa) podrá cargarse escribiendo un byte en cualquier dirección del rango:

001X.XXX1.01XX.XXXX.XXXX.XXXX.XXXX
nGCS1 CS6

- Aprovecharemos que la configuración básica del SoC está hecha por el programa residente en ROM que se ejecuta tras reset.
  - Pilas, interrupciones, memoria, de puertos de E/S, de reloj, etc...
  - Si no existiese tal programa, la aplicación debería configurarlo todo.

# Programa principal

## alternativa 1

```
#define SEGS (*(volatile unsigned char *)...)

void main( void )
{
  unsigned char i;
  unsigned int j;

  SEGS = 0xff;
  while( 1 )
    for( i=0; i<16; i++ ) {
      for( j=0; j<300000; j++ );
      switch( i ){
        case 0x0:
          SEGS = 0x12; break; // 0b000100010
        case 0x1:
          SEGS = ...; break;
        ...
        case 0xf:
          SEGS = ...; break;
        default:
          SEGS = ...; break;
      }
    }
}
```

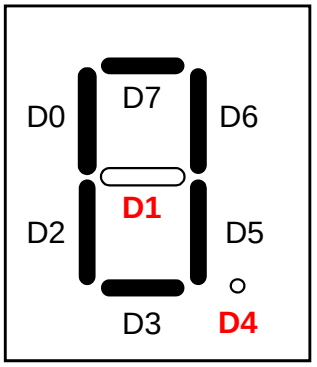
declaración del nemotécnico para la dirección de escritura en el display

declara variables índice

apaga todos los segmentos escribiendo 0xff en la dirección en donde se mapea el display

pequeño retardo

visualiza el 0




indefinidamente visualiza los números del 0x0 al 0xF

# Programa principal

## alternativa 2

- Una solución más compacta sería:

```
#define SEGS (*(volatile unsigned char *)...)

const unsigned char hex2segs[16] = {0x12, ...}

void main( void )
{
  unsigned char i;
  unsigned int j;

  SEGS = 0xff;
  while( 1 )
    for( i=0; i<16; i++ ) {
      for( j=0; j<300000; j++ );
      SEGS = hex2segs[i];
    }
}
```

contiene los patrones de segmentos de cada dígito hexadecimal

- En cada caso, el programa ocupa:
  - 464 B: primera alternativa con switch
  - 136+16 B: segunda alternativa con array

# Tareas

1. Copiar el proyecto **`lab0`** en la vista del Project Explorer:
   - Renombrarlo **`lab1`**
   - Eliminar del directorio del proyecto el fichero **`lab0.c`**
   - Renombrar el fichero **`lab0.ld`** como **`lab1.ld`**
   - Indicar en los settings del enlazador que use el fichero **`lab1.ld`**
2. Descargar de la Web el fichero **`lab1.c`** en el directorio del proyecto.
3. Refrescar proyecto.
4. Completar el código omitido en el fichero **`lab1.c`**
5. Compilar proyecto.
6. Crear un duplicado de la configuración de depuración **`lab0`**
   - Renombrarla **`lab1`**
   - Usar en ella como proyecto **`lab1`** y como aplicación **`Debug\lab1.elf`**
7. Conectar la placa y encenderla.
8. Arrancar OpenOCD.
9. Arrancar la configuración de depuración **`lab1`**