**Tema 2: Representación de la información**

1. *Operaciones lógicas*

* Se aplican bit a bit
* Se usan las siguientes tablas:

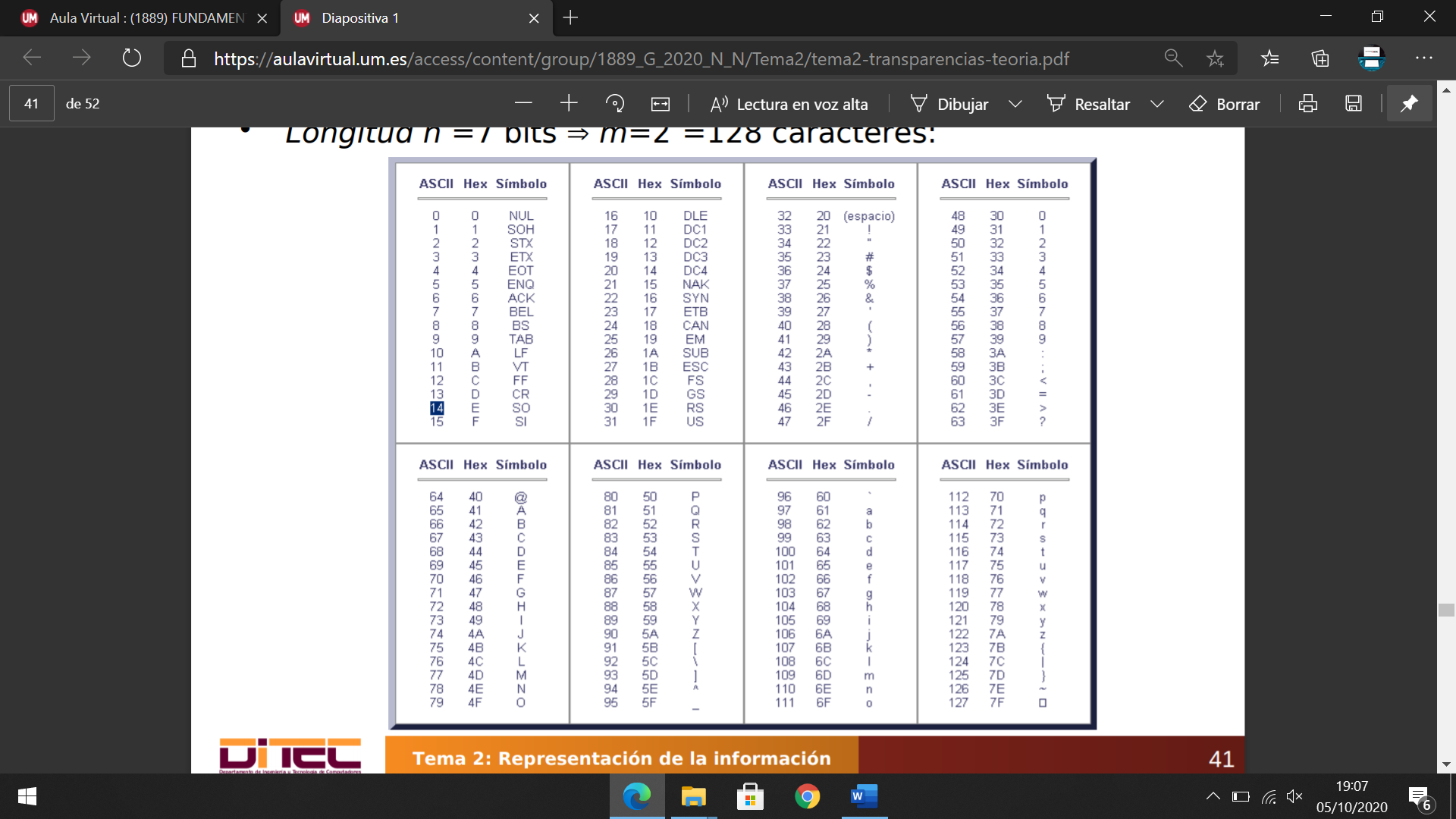
|  |  |  |
| --- | --- | --- |
| Suma Lógica (OR) | Producto Lógico (AND) | Complemento (NOT) |
| 0 + 0 = 0 | 0 · 0 = 0 | 0 = 1 |
| 0 + 1 = 1 | 0 · 1 = 0 | 1 = 0 |
| 1 + 0 = 1 | 1 · 0 = 0 |
| 1 + 1 = 1 | 1 · 1 = 1 |
| * Representación grafica de las operaciones (Puertas lógicas) |  |

**2.5.** *Representación de caracteres*

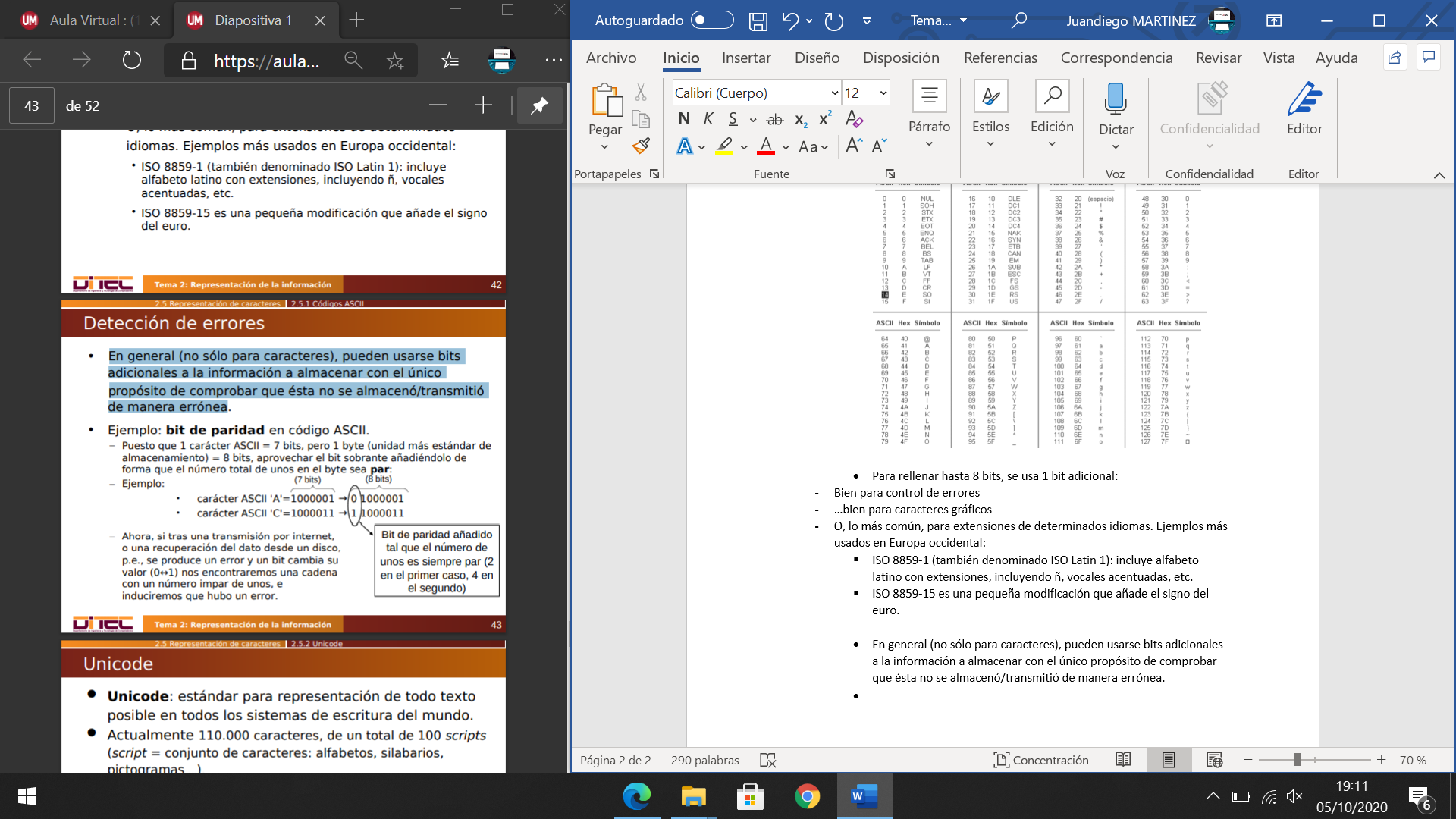
* **Códigos de Entrada/Salida:**
* Los códigos de E/S (o externos): asocian a cada carácter una determinada combinación de bits
* Correspondencia entre x = {A..Z,a..z,0..9,(,),\*…} y B = {0,1}n
* El |x| = m dependerá del dispositivo de E/S o del sistema
* Necesitaremos un numero de bits n tal que: n >= log2m
* Ejemplo: para codificar {0,1,…,9} se necesitarán 4 bits.
* La elección del código es arbitraria, pero existen códigos normalizados.
* El más popular es el código ASCII (American Standard Code for Information Interchange)
* Codifica, desde los años 60, la inmensa mayoría de caracteres impresos usados en textos en idioma inglés.

***Código ASCII***

* Longitud n=7 bits => m=27=128 caracteres:

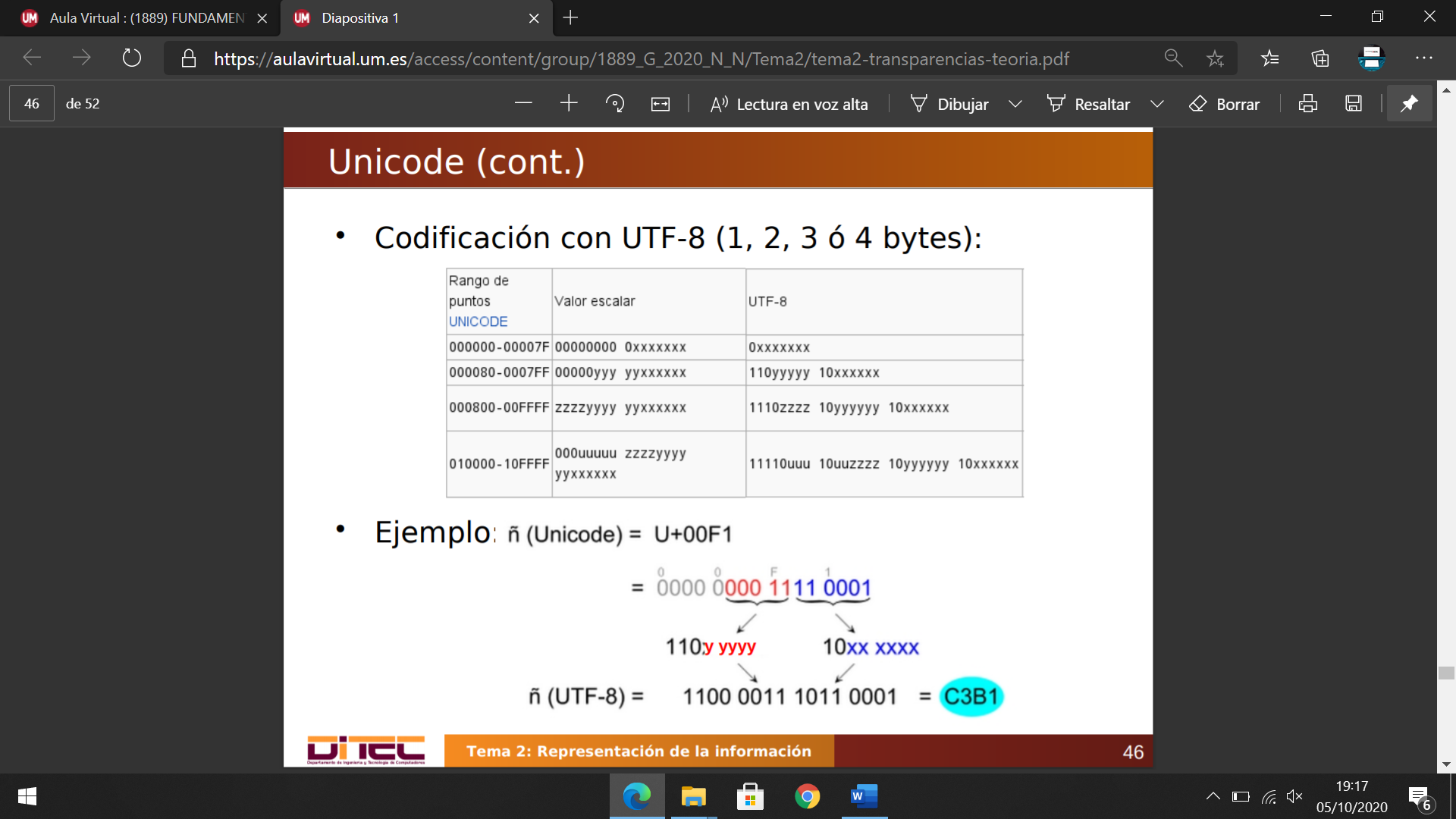


* Para rellenar hasta 8 bits, se usa 1 bit adicional:
* Bien para control de errores
* …bien para caracteres gráficos
* O, lo más común, para extensiones de determinados idiomas. Ejemplos más usados en Europa occidental:
* ISO 8859-1 (también denominado ISO Latin 1): incluye alfabeto latino con extensiones, incluyendo ñ, vocales acentuadas, etc.
* ISO 8859-15 es una pequeña modificación que añade el signo del euro.
* En general (no sólo para caracteres), pueden usarse bits adicionales a la información a almacenar con el único propósito de comprobar que ésta no se almacenó/transmitió de manera errónea.



***Unicode***

* **Unicode**: estándar para representación de todo texto posible en todos los sistemas de escritura del mundo.
* Actualmente 110.000 caracteres, de un total de 100 scripts (script = conjunto de caracteres: alfabetos, silabarios, pictogramas …).
* Inicialmente, era de 16 bits (216=65.536 posibilidades). Actualmente, Unicode define un espacio de of 1.114.112 code points en el rango 0)16 a 10FFFF)16.
* Un "code point" se expresa como U+número (p.e. U+0058
* para la 'X', o U+6708 para el ideograma chino "yue": ). (Cada carácter o símbolo se define por un nombre e identificador numérico, llamado code point)
* Para "code points" que no caben en 16 bits (fuera del llamado Basic Multilingual Plane, o BMP), se usan cinco o seis dígitos, según se requiera.
* Los primeros 256 "code points" son idénticos a la codificación ISO-8859-1 (Latin 1), para facilitar la conversión, y compatibilizar el subconjunto de caracteres más utilizado de largo.
* Los "code points" pueden concretarse en secuencias de bytes de varias formas, llamadas "encodings". Las más comunes son UTF8 y UTF-16 (UTF = Unicode Transformation Format), aunque hay más (UCS-2, UCS-4, etc.). (UTF-8 usa un byte por cada carácter ASCII (→ compatibilidad). Sólo para caracteres no ASCII empiezan a utilizarse dos o más bytes (hasta cuatro), según un esquema basado en la comprobación de los primeros bits de cada byte, que indican si el byte siguiente forma parte aún del mismo "code point" (ilustración en transparencia siguiente). (– UTF-16 similar, pero usando como base ristras de 16 bits (dos bytes), con unos pocos caracteres usando dos unidades de 16 bits (e.d. 4 bytes). Menos usado por desperdiciar más espacio).



**2.6.** *Representación de imágenes*

* **Mapas de bits:**
* Las imágenes están compuestas por infinitos puntos
* Cada punto tiene un nivel de gris o un color
* Para codificar una imagen:
* Hay que determinar cuántos puntos vamos a considerar (no podemos almacenar y procesar infinitos puntos).
* Hay que asociarles un nivel de color o de gris (el nivel asociado será la media de los puntos representados)
* Consideraremos que una imagen está formada por una matriz de píxeles (elementos de imagen, picture elements)
* Resolución de la imagen (número de elementos por línea \* número de elementos por columna)
* Representación de la imagen (sencilla, funciona con cualquier imagen) (se almacena sucesivamente los atributos de los píxeles, como por ejemplo de izquierda a derecha y de arriba abajo)
* **Mapas de vectores:**
* Representación como colección de objetos (líneas, polígonos, textos)
* Los objetos se modelan mediante vectores y ecuaciones
* Al visualizar la imagen en pantalla se evalúan las ecuaciones y se escalan los vectores para generar la imagen
* Características de este tipo de imágenes:
* Son adecuadas para gráficos geométricos, pero no para imágenes reales
* Son ideales para aplicaciones de diseño por ordenador (CAD)
* En comparación con los mapas de bits, los archivos de imágenes vectoriales suelen ocupar mucho menos espacio
* Es más fácil procesar o escalar las imágenes a cualquier tamaño
* La fidelidad o calidad de la imagen respecto a la realidad suele ser peor

