Instrucciones del 8088/8086.

Instrucciones de trasferencia de datos.

- **Nombre:** **MOV**
- **Formato:** **MOV destino, origen**
- **Descripción:**
  Transfiere un byte o una palabra desde el operando origen al operando destino.

- **Nombre:** **PUSH**
- **Formato:** **PUSH origen**
- **Descripción:**
  Decrementa el puntero de pila (SP) en 2 y luego transfiere la palabra que se ha especificado en el operando origen a lo alto de la pila.

- **Nombre:** **POP**
- **Formato:** **POP destino**
- **Descripción:**
  Transfiere un byte o una palabra desde la cima de la pila al operando destino y luego incrementa la pila en 2.

Instrucciones de bifurcación.

- **Nombre:** **CALL**
- **Formato:** **CALL destino**
- **Descripción:**
  Bifurca a un procedimiento, salvando antes la dirección de la instrucción siguiente en la pila para poder volver a dicha instrucción una vez ejecutado el procedimiento.
  El procedimiento puede estar dentro del mismo segmento (llamada NEAR) o en otro segmento (llamada FAR).

- **Nombre:** **RET**
- **Formato:** **RET**
- **Descripción:**
  Retorna de un procedimiento a la dirección salvada en la pila. Dependiendo de si se vuelve de un procedimiento NEAR o de un procedimiento FAR el retorno se hace de forma diferente. En el primer caso, se quita de la cima de la pila una palabra que corresponde a la dirección de retorno. En el caso de un procedimiento FAR se quitan dos palabras, la primera se corresponde con el desplazamiento y las segunda al segmento de la dirección de retorno.

- **Nombre:** **INT**
- **Formato:** **INT tipo_interrupción**

- **Descripción:**
  INT activa el procedimiento de interrupción especificado por el operando. La dirección del vector de interrupción se calcula multiplicando por 4 el operando, que es un valor entre 0 y 255.
  El vector de interrupción se compone de dos palabras: la primera palabra es el desplazamiento y la segunda el segmento.

- **Nombre:** **IRET**
- **Formato:** **IRET**
- **Descripción:**
  Devuelve el control a la dirección de retorno salvada en la pila y restaura los flags. Se emplea para finalizar un procedimiento de interrupción.

- **Nombre:** **LOOP**
- **Formato:** **LOOP desplazamiento**
- **Descripción:**

  Si CX es diferente de cero, entonces IP = IP + desplazamiento. Si CX es cero entonces ejecuta la instrucción siguiente. El desplazamiento debe estar comprendido entre -128 y 127.

  Mediante esta instrucción es posible implementar bucles. También son factibles los bucles anidados pero debemos hacer uso de la pila.

- **Nombre:** **JMP**
- **Formato:** **JMP dirección.**
- **Descripción:**

  Realiza un salto incondicional. La bifurcación puede ser dentro del mismo segmento, en cuyo caso IP se sustituye por el valor del desplazamiento. Si la bifurcación es a otro segmento se sustituyen los valores correspondientes a CS y a IP.

  La bifurcación puede ser especificando una etiqueta o la dirección.

- **Nombre:** **J XXX**
- **Formato:** **J {condición de salto} dirección**
- **Descripción:**

  Salta a la dirección especificada en el operando si es cierta la condición. En caso de no satisfacerse la condición se ejecuta la instrucción siguiente. (* comparación sin signo)

  Las condiciones de salto son:

Instrucción	Bifurca si la condición es:
JA*	superior
JAE*	superior o igual
JB*	inferior
JBE*	inferior o igual
JC	acarreo
JCXZ	CX = 0
JE	igual
JG	mayor
JGE	mayor o igual
JL	menor
JLE	menor o igual
JNA*	no superior
JNAE*	no superior ni igual
JNB*	no inferior
JNBE*	no inferior no igual
JNC	no acarreo
JNE	no igual
JNG	no mayor
JNGE	no mayor ni igual
JNL	no menor
JNLE	no menor o igual
JNO	no desbordamiento
JNP	no paridad
JNS	no signo(positivo)
JNZ	no cero
Instrucción	Bifurca si la condición es:
JO	overflow
JP	paridad
JPE	paridad par
JPO	paridad impar
JS	signo (negativo)
JZ	cero

Instrucciones aritméticas.

- **Nombre:** **ADD**
- **Formato:** **ADD destino, origen**
- **Descripción:**

  Suma los dos operandos y el resultado lo deja en el operando destino. Los operandos deben ser del mismo tipo.

- **Nombre:** **ADC**
- **Formato:** **ADC destino, origen**
- **Descripción:**

  Suma los dos operandos. Suma uno si estás activado el flag de acarreo. El resultado se almacena en el operando destino. Además los operandos deben ser del mismo tipo.

- **Nombre:** **SUB**
- **Formato:** **SUB destino, origen**
- **Descripción:**

  Resta el operando origen del operando destino. El resultado se almacena en el operando destino y además, ambos operandos deben ser del mismo tipo.

- **Nombre:** **SBB**
- **Formato:** **SBB destino, origen**
- **Descripción:**

  Resta el operando origen del operando destino. Resta uno si el flag de acarreo está activo. Los operandos deben ser del mismo tipo. El resultado se almacena en el operando destino.

- **Nombre:** **MUL**
- **Formato:** **MUL origen**
- **Descripción:**

  Multiplica, sin considerar el signo el acumulador (AL o AX) por el operando origen. Si el operando origen es de tipo byte el resultado se almacena en AX. Si es de tipo palabra se almacena en DX (palabra superior) y AX (palabra inferior).

- **Nombre:** **IMUL**
- **Formato:** **IMUL origen**
- **Descripción:**

  Multiplica, considerando el signo, el acumulador AL o AX por el operando origen. Si el operando fuente es un byte se almacena el resultado en AX. Si se trata de una palabra, se almacena en DX (palabra superior) AX (palabra inferior).

- **Nombre:** **DIV**
- **Formato:** **DIV origen**
- **Descripción:**

  Divide, sin considerar el signo, el acumulador AL o AX y su extensión (AH o DX) por el operando origen. El resultado se almacena en AL o AX, según el operando sea de un byte o de una palabra. El resto se almacena en la extensión del acumulador AH o DX.

- **Nombre:** **IDIV**
- **Formato:** **IDIV origen**
- **Descripción:**

  Divide, considerando el signo, el acumulador AL o AX y su extensión (AH o DX) por el operando origen. El resultado se almacena en AL o AX, según el operando sea de un byte o de una palabra. El resto se almacena en la extensión del acumulador AH o DX.

- **Nombre:** **INC**
- **Formato:** **INC destino**
- **Descripción:**

  Suma una unidad al operando destino. El operando puede ser de tipo byte o palabra.
- **Nombre:** **DEC**
- **Formato:** **DEC destino**
- **Descripción:**

  Resta una unidad al operando destino. El operando puede ser de tipo byte o palabra.

- **Nombre:** **NEG**
- **Formato:** **NEG destino**
- **Descripción:**

  Cambia de signo mediante el complemento a 2 del operando destino. Deja el resultado en el operando destino. El operando puede ser de tipo byte o palabra.

Instrucciones de comparación.

- **Nombre:** **CMP**
- **Formato:** **CMP destino, origen**
- **Descripción:**

  Resta el operando origen del operando destino pero no devuelve el resultado. Los operandos son del tipo byte o palabra. Se alteran los flag dependiendo del resultado.

Instrucciones lógicas.

- **Nombre:** **AND**
- **Formato:** **AND destino, origen**
- **Descripción:**

  Es una operación Y lógica a nivel de bits. El resultado se almacena en destino.

A	B	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

- **Nombre:** **OR**
- **Formato:** **OR destino, origen**
- **Descripción:**

  Es una operación O lógica a nivel de bits. El resultado se almacena en destino.

A	B	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

- **Nombre:** **OR**
- **Formato:** **XOR destino, origen**
- **Descripción:**

  Es una operación O lógica EXCLUSIVO a nivel de bits. El resultado se almacena en destino.

A	B	a OR b
0	0	0
0	1	1
1	0	1
1	1	0

- **Nombre:** **NOT**
- **Formato:** **NOT destino**
- **Descripción:**

  Realiza el complemento a 1.

A	NOT a
0	1
1	0

Instrucciones de desplazamiento.

- **Nombre:** **SAR**
- **Formato:** **SAR destino, contador**
- **Descripción:**

  Desplaza a la derecha los bits del operando destino el número de bits especificado en contador. Si el número de bits a desplazar es 1 se puede especificar directamente. Si es mayor que uno **se debe cargar el valor en CL**.

- **Nombre:** **SAL**
- **Formato:** **SAL destino, contador**
- **Descripción:**

  Desplaza a la izquierda los bits del operando destino el número de bits especificado en contador. Si el número de bits a desplazar es 1 se puede especificar directamente. Si es mayor que uno **se debe cargar el valor en CL**.

- **Nombre:** **SHR**
- **Formato:** **SHR destino, contador**
- **Descripción:**

  Desplaza a la derecha de manera lógica los bits del operando destino el número de bits especificado en contador. Si el número de bits a desplazar es 1 se puede especificar directamente. Si es mayor que uno **se debe cargar el valor en CL**.

- **Nombre:** **SHL**
- **Formato:** **SHL destino, contador**
- **Descripción:**

  Desplaza a la izquierda de forma lógica los bits del operando destino el número de bits especificado en contador. Si el número de bits a desplazar es 1 se puede especificar directamente. Si es mayor que uno **se debe cargar el valor en CL**.

- **Nombre:** **ROR**
- **Formato:** **ROR destino, contador**
- **Descripción:**

  Rota a la derecha los bits del operando destino el número de bits especificado en contador. Si el número de bits a desplazar es 1 se puede especificar directamente. Si es mayor que uno **se debe cargar el valor en CL**.

- **Nombre:** **ROL**
- **Formato:** **ROL destino, contador**
- **Descripción:**

  Rota a la izquierda los bits del operando destino el número de bits especificado en contador. Si el número de bits a desplazar es 1 se puede especificar directamente. Si es mayor que uno **se debe cargar el valor en CL**.

- **Nombre:** **RCR**
- **Formato:** **RCR destino, contador**
- **Descripción:**

  Rota a la derecha los bits del operando destino y el flag de acarreo el número de bits especificado en contador. Si el número de bits a desplazar es 1 se puede especificar directamente. Si es mayor que uno **se debe cargar el valor en CL**.

- **Nombre:** **RCL**
- **Formato:** **RCL destino, contador**
- **Descripción:**

  Rota a la izquierda los bits del operando destino y el flag de acarreo CF el número de bits especificado en contador. Si el número de bits a desplazar es 1 se puede especificar directamente. Si es mayor que uno **se debe cargar el valor en CL**.

Instrucciones de bit.

- **Nombre:** **TEST**
- **Formato:** **TEST destino, fuente**
- **Descripción:**

  Es igual que la operación AND (y lógico) pero no se guarda el resultado en destino. Se realiza a nivel de bit y modifica los flag de estado

- **Nombre:** **CLI**
- **Formato:** **CLI**
- **Descripción:**

  Desactiva las interrupciones. Las interrupciones no enmascarables no se pueden inhibir.

- **Nombre:** **STI**
- **Formato:** **STI**
- **Descripción:**

  Permite las interrupciones.

- **Nombre:** **CLC**
- **Formato:** **CLC**
- **Descripción:**

  Desactiva el flag de acarreo.

- **Nombre:** **STC**
- **Formato:** **STC**
- **Descripción:**

  Pone a 1 el flag de acarreo

Instrucciones de entrada /salida.

- **Nombre:** **IN**
- **Formato:** **IN acumulador, puerta**
- **Descripción:**

  Transfiere un byte o una palabra de una puerta de entrada del procesador al registro AL o AX. El número de la puerta se puede especificar de las formas siguientes:

  - Un valor variable almacenado en el registro DX
  - Un valor fijo entre 0 y 255.

- **Nombre:** **OUT**
- **Formato:** **OUT acumulador, puerta**
- **Descripción:**

  Transfiere un byte o una palabra A una puerta de entrada del procesador desde el registro AL o AX. El número de la puerta se puede especificar de las formas siguientes:

  - Un valor variable almacenado en el registro DX
  - Un valor fijo entre 0 y 255.

Instrucciones varias.

- **Nombre:** **WAIT**
- **Formato:** **WAIT**
- **Descripción:**

  Permite la comunicación con otros coprocesadores. Pone al procesador en un estado de espera hasta que se activa la línea TEST. La instrucción WAIT comprueba la línea TEST cada cinco intervalos de reloj.

- **Nombre:** **HLT**
- **Formato:** **HLT**
- **Descripción:**

  Para el procesador. Volverá a trabajar si:
  - Se hace RESET en el ordenador.
  - Se recibe una interrupción no enmascarable.
  - Se recibe una interrupción enmascarable siempre que se permitan las interrupciones, es decir, que previamente no las hayamos deshabilitado con CLI.

- **Nombre:** **NOP**
- **Formato:** **NOP**
- **Descripción:**

El procesador no hace nada y pasa a ejecutar la instrucción siguiente.

DIRECTIVAS.

Directivas de datos.

- **Nombre:** **EQU**

- **Formato:** **nombre EQU expresión**
- **Descripción:**

  Asigna a nombre el valor de una expresión, que puede ser:
  - Una constante numérica.
  - Otro nombre.
  - Cualquier operación con números y nombres que de un resultado numérico.
  - Una referencia a una dirección empleando cualquier tipo de direccionamiento.

Nombre: **DB**
- **Formato:** **nombre DB expresión [, ...]**
- **Descripción:**

  Reserva memoria para una variable de 8 bits, nombre es opcional y únicamente identifica el primer byte.

- **Nombre:** **DD**
- **Formato:** **nombre DD expresión [, ...]**
- **Descripción:**

  Reserva memoria para una variable de tipo doble palabra, es decir 4 bytes ó 32 bits, nombre es opcional y únicamente identifica el primer byte.

- **Nombre:** **DQ**
- **Formato:** **nombre DQ expresión [, ...]**
- **Descripción:**

  Reserva memoria para una variable de tipo cuádruple palabra, es decir 64 bits, nombre es opcional y únicamente identifica el primer byte.

- **Nombre:** **DT**
- **Formato:** **nombre DT expresión [, ...]**
- **Descripción:**

  Reserva memoria para una variable de tipo diez bytes de memoria, es decir 80 bits, nombre es opcional y únicamente identifica el primer byte.

- **Nombre:** **DW**
- **Formato:** **nombre DW expresión [, ...]**
- **Descripción:**

  Reserva memoria para una variable de tipo palabra, es decir 2 bytes ó 16 bits, nombre es opcional y únicamente identifica el primer byte.

- **Nombre:** **PUBLIC**
- **Formato:** **PUBLIC símbolo [, ...]**
- **Descripción:**

  Permite que los símbolos especificados sean accesibles por otros módulos.

- **Nombre:** **EXTRN**
- **Formato:** **EXTRN nombre:tipo [, ...]**
- **Descripción:**

  Sirve para poder hacer referencia a símbolos definidos en otros módulos.

- **Nombre:** **END**
- **Formato:** **END [expresión]**
- **Descripción:**

  Indica el final del programa fuente. La expresión es una etiqueta que sirve para identificar el comienzo del programa fuente.

- **Nombre:** **SEGMENT**
- **Formato:**

  nombre SEGMENT [alineamiento] [combinación] [‘clase’]
- **Descripción:**

  Indica el comienzo del segmento de datos “nombre”. El nombre de ambas directivas debe ser el mismo.

  Un segmento es un bloque que puede contener las sentencias:
  - Definición de variables.
  - Instrucciones.
  - Combinación de las dos anteriores.

  Un módulo fuente en ensamblador puede ser:
  - Parte de un segmento.
  - Un segmento.
  - Partes de varios segmentos.
  - Varios segmentos.
  - Combinaciones de los anteriores.

- **Nombre:** **ENDS**
- **Formato:** **nombre_segmento ENDS**
  nombre_estructura ENDS
- **Descripción:**
  Indica el final del segmento o de la estructura.

- **Nombre:** **ASSUME**
- **Formato:** **ASSUME reg_seg:nombre [, ...]**
- **Descripción:**
  Indica al ensamblador el registro de segmento que se va a utilizar para direccionar cada segmento dentro del módulo.
  Los registros de segmentos son:
  - CS: segmento de código.
  - DS: segmento de datos.
  - SS: segmento de pila.
  - ES: segmento extra de datos.

- **Nombre:** **PROC**
- **Formato:** **nombre PROC [atributo]**
- **Descripción:**

Indica el comienzo del procedimiento nombre. Un procedimiento es un bloque de instrucciones.

- **Nombre:** **ENDP**
- **Formato:** **nombre ENDP**
- **Descripción:**
  Indica el final del procedimiento.

- **Nombre:** **GROUP**
- **Formato:** **nombre GROUP nom_seg[, ...]**
- **Descripción:**
  Agrupa dos o más segmentos lógicos en un único segmento físico con el nombre NOMBRE.

- **Nombre:** **RECORD**
- **Formato:**

nombre RECORD campo:longtud [=exp][, ...]

- **Descripción:**
  Define una plantilla de campos a nivel de bits pero no reserva memoria para ella,
  Cada campo se define mediante un nombre, una longitud en bits y un valor opcional que es el valor por omisión.

- **Nombre:** **STRUCT**
- **Formato:** **nombre STRUCT**
- **Descripción:**
  Define una plantilla de campos a nivel de bytes. Una vez definida la estructura, su nombre se puede usar para reservar e inicializar memoria.

Directivas condicionales.

- **Nombre:** **IF**
- **Formato:**

```
IFxxx [condición ]        IFxxx [condición ]
    ....                      ....
    ....                      ....
    ENDIF                 ELSE
                              ....
                              ....
                              ENDIF
```

- **Descripción:**

  Las directivas condicionales sirven para que el ensamblador incluya u omita ciertas partes del programa fuente. Si la condición se cumple, se incluye la parte de programa comprendida entre las sentencias IF y ELSE o IF y ENDIF si no hay rama ELSE.

  Si la conclusión no se cumple o no se incluye nada o se tiene en cuenta la parte comprendida entre el ELSE y el ENDIF.

  Las directivas condicionales de compilación son:

Directiva	Condición
IF expresión	la expresión es distinta de cero.
IFE expresión	la expresión es cero.
IF1	primer paso del ensamblador.
IF2	segundo paso del ensamblador.
IFDEF símbolo	símbolo definido o declarado externo.
IFNDEF símbolo	símbolo no se ha definido ni declarado externo.
IFB argumento	el argumento está en blanco.
IFNB argumento	el argumento no está en blanco.
IFIDN arg1, arg2	arg1 es idéntico a arg2.
IFIDF arg1, arg2	arg1 es diferente a arg2.

Directivas de listados.

- **Nombre:** **COMMENT**
- **Formato:** **COMMENT delimitador texto delimitador**
- **Descripción:**

  Permite insertar comentarios en el programa sin tener que especificar el carácter "," en cada línea.

  Delimitador es el primer carácter diferente a blanco que aparece. Ambos delimitadores deben ser iguales.

- **Nombre:** **%OUT**
- **Formato:** **%OUT mensaje**
- **Descripción:**

  Esta directiva sirve para escribir un mensaje por pantalla cuando pasamos el ensamblador al programa. Se suele utilizar para indicar que camino ha sido elegido en una determinada condición.

- **Nombre:** **PAGE**
- **Formato:** **PAGE [operando1] , [operando2]**
- **Descripción:**

  Se emplea para definir la longitud y ancho de la página en el listado del programa, así como para definir los capítulos.

  El operando1 puede ser:

  - Número de líneas por página de 10 a 255. Por omisión toma el valor de 66.
  - El signo +, indica nuevo capítulo.

  El operando2 es:

  - La anchura de la línea va desde 60 a 132. Por omisión vale 80.

- **Nombre:** **TITTLE**
- **Formato:** **TITTLE texto**
- **Descripción:**

  Especifica el título que aparecerá en el listado como primera línea en cada página. La longitud máxima del título es de 60 caracteres.

Funciones de la interrupción 21h.

La tabla siguiente representa las funciones más empleadas de la interrupción 21h.

Función	Descripción
AH = 1h	**Entrada desde el teclado**. Esta función espera a que se teclee un carácter por teclado. Escribe el carácter en pantalla y devuelve el código ASCII en el registro AL
	Modifica **AL** con el código ASCII del carácter leído.
AH = 2h	**Salida a la pantalla**. Muestra un carácter en pantalla. Se debe guardar en **DL** el código ASCII del carácter que se desea sacar por pantalla.
	Modifica **DL** se le asigna el código ASCII del carácter que se desea sacar por pantalla.
	Modifica **AL** devuelve un código de error.
AH = 8h	**Entrada desde el teclado sin reproducir el carácter por la pantalla**. Lee un carácter por pantalla pero no lo muestra por pantalla.
	Modifica **AL** con el código ASCII del carácter leído.
AH = 9h	**Muestra cadena**. Muestra por pantalla la cadena a la que apunta la pareja de registros DS:DX. El final de la cadena se debe marcar con el carácter $.
AH = 0Ah	**Lee cadena**. Lee una cadena desde el teclado.
AH = 4Ch	**Sale al DOS.** Devuelve el control al DOS, igual que la interrupción 20h. Devuelve en **AL** el código de retorno al DOS que se desee.
	Modifica **AL** con el valor que se desea devolver al DOS para ser usado con el IF ERRORLEVEL.

Funciones de la interrupción 10h

.

AH	Función.
00h	Establece el modo de la pantalla. AL = 0 40 x 25 blanco y negro alfanumérico. AL = 4 320 x 200 color gráfica. AL = 1 40 x 25 color alfanumérico. AL = 5 320 x 200 blanco y negro gráfica AL = 2 80 x 25 blanco y negro alfanumérico. AL = 6 640 x 200 blanco y negro gráfica AL = 3 80 x 25 color alfanumérico.
01h	Establecer las líneas del cursor. CH (bits 0-4) Línea inicial. CL (bits 0-4) Línea final. CH (bits 5-7) deben ser 0. CL (bits 5-7) deben ser 0.
02h	Posición del cursor. DH = Fila (0-24) DL = Columna (0-79) BH= número de página
03h	Leer posición del cursor. BH = número de página (0 en modo gráfico) Devuelve: DH (fila) DL (columna) CH (bits 0-4) línea inicial CH(bits 5-7) 0 CL (bits 0-4) línea final y CL (bits 5-7) 0
06h	Desplazamiento (*scroll*) hacia arriba. AL es el número de líneas. Si AL=0 se borra la ventana. CH fila esquina superior izquierda. DH fila esquina inferior derecha. BH relleno. CL columna esquina superior izquierda. DL columna esquina inferior derecha.

AH	Función.
07h	Desplazamiento (*scroll*) hacia arriba. AL es el número de líneas. Si AL=0 se borra la ventana. CH fila esquina superior izquierda. DH fila esquina inferior derecha. BH relleno. CL columna esquina superior izquierda. DL columna esquina inferior derecha.
08h	Leer carácter y atributo de la posición actual BH es el número de página. Devuelve: AL el carácter leído AH atributo del carácter leído.
09h	Escribir el carácter y el atributo en la posición actual del cursor. BH es el número de página. CX es el número de caracteres a escribir. BL es el atributo del carácter o el color. AL es el carácter a escribir.
0Ah	Escribir un carácter en la posición actual del cursor AL = carácter BL= color BH = número de página CX = número de caracteres a escribir
0Eh	Escribir el carácter en la pantalla y avanzar el cursor. AL = carácter a escribir. BL = color del carácter o su atributo. BH = número de la página.
0Fh	Leer el estado actual de la pantalla. Devuelve: AL el modo. AH número de columnas de la pantalla. BH número de página activa.

Tabla ASCII en decimal.

850 Multilingüe (Latín 1)

0		32		64	@	96	`	128	Ç	160	á	192	└	224	Ó
1	☺	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	☻	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Ô
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
4	♦	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
6	♠	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
7	•	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	Þ
8	◘	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	þ
9	○	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
10	◙	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
11	♂	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
12	♀	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
13	♪	45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
14	♫	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
15	☼	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
16	►	48	0	80	P	112	p	144	É	176	░	208	ð	240	
17	◄	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
18	↕	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
19	‼	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
20	¶	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
21	§	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
22	▬	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
23	↨	55	7	87	W	119	w	151	ù	183	À	215	Î	247	¸
24	↑	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	←	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
28	∟	60	<	92	\	124	¦	156	£	188	╝	220	▄	252	³
29	↔	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
30	▲	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
31	▼	63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	