Arellano Granados Angel Mariano

218123444

Seminario de Traductores de Lenguajes I

I7026 D02

Reporte de Actividades 3

Actividades 3 – Parte 1 – 3

Actividad 3 – Parte 1

Descripción

Contestar el cuestionario.

Desarrollo y Resultados

Actividad 2 – Parte III

Entorno EMU8086

Nombre:	Angel Mariano Arellano Granados	Código	218123444	Sección:	D-02

Entre las opciones de la barra de herramientas, se localiza una opción que permite activar los dispositivos virtuales para simular los puertos físicos ¿Cuál es el nombre de esta opción?					
	External	X	Virtual devices		Virtual drive
	Debug		Ninguno.		
La ventana principal del editor de texto cuenta con una barra de menú de Windows con la opción <assembler>					
	Falso.	X	Verdadero.		No lo sé.
Los botones <aux> + <memory> que se habilitan con la opción "emulate" permiten ver el mapa de memoria.					
X	Falso.		Verdadero.		No lo sé.
La opción se llama <view> + <memory>					
Es una opción del entorno EMU86 que permite llamar al browser y explorar una gran variedad de documentos de ayuda.					
	New	X	Code examples		Quick start tutor
	Recent file		Ninguno.		
El botón <view>, se localiza una opción denominada "options" que permite ver el contenido de los registros en distintas bases (hexadecimal, octal, ASCII, etc).					
X	Falso.		Verdadero.		No lo sé.
En el manejo de interrupciones, el EMU8086, verifica continuamente en el bit <IF> del registro "flags" si se presentó alguna, transfiriendo el control a la subrutina de atención de estas.					
	Falso.	X	Verdadero.		No lo sé.
Por defecto, las interrupciones por hardware se encuentran habilitadas y cuando se produce una interrupción de este tipo, el EMU8086 lo indica con la leyenda ::"hardware interrupt".					
	Falso.	X	Verdadero.		No lo sé.
En el menú debug se puede insertar un "break point" cuando se está depurando el programa.					

	Falso.	X	Verdadero.		No lo sé.

Dentro del entorno del editor se localiza un botón para habilitar la opción del emulador y el despliegue del código fuente ¿Cuál es la opción que muestra esa opción?

	<compile>		<converter>	X	<emulate>
	<calculator>		Ninguno.		

¿Cuál de las siguientes opciones de la barra de herramientas del menú principal del EMU8086 habilita la documentación de su entorno?

	<open>		<help>		<options>
X	<about>		Ninguno.		

Es una opción del entorno EMU8086 que permite escribir un nuevo código en lenguaje ensamblador con extensión ASM.

X	New		Code examples		Quick start tutot
	Recent file		Ninguno.		

Cuando se produce una interrupción en el CPU, se detiene la ejecución del programa que está procesando y este procede a ejecutar un subprograma asociado a la interrupción. Para ello, el CPU actualiza la dirección en sus registros CS:IP, a fin de actualizar su dirección de memoria para ejecutar el subprograma ubicado en la nueva dirección.

	Falso.		Verdadero.	X	No lo sé.

En el manejo de una interrupción, el CPU a través de los registros CS:IP se ubica en una nueva dirección para ejecutar el subprograma asociado a esta. La nueva dirección se extrae de memoria en una "Tabla de vectores de interrupción" que tiene 512 posiciones con valores CS e IP de 16 bits para cada uno.

	Falso.		Verdadero.	X	No lo sé.

En caso de elegir la opción <New> en el entorno del EMU8086 ¿Cuál es la opción a elegir para crear un programa ejecutable simple, pero con valores predefinidos para ubicar el código?

	BIN template		COM template		BOOT template
X	EXE template		Ninguno.		

En caso de elegir la opción <New> en el entorno del EMU8086 ¿Cuál es la opción a elegir para crear un programa ejecutable avanzado sin limitaciones de tamaño, ni de segmentos?

X	BIN template		COM template		BOOT template
	EXE template		Ninguno.		

¿Cuál es la opción que permite ver el estado del registro de banderas en el procesador?

	stack	X	flags		options
	Symbol table		Ninguno.		

Entre las opciones de la barra de herramientas, se localiza la opción "Debug", la cual provee herramientas para depurar los programas.

	Falso.	X	Verdadero.		No lo sé.

Las interrupciones por hardware se encuentran deshabilitadas cuando el bit IF=1.

	Falso.	X	Verdadero.		No lo sé.
La opción de <NEW> que crea archivos ejecutables con formato simple y permite un offset (desplazamiento) de 256 bytes. Sus códigos inician con la directiva ORG 100h (DOS y Windows)					
	BIN template	X	COM template		BOOT template
	EXE template		Ninguno.		
LOAD, RELOAD, SINGLE STEP, STEP BACK y RUN, son botones que se encuentran bajo la barra de herramientas.					
	Falso.	X	Verdadero.		No lo sé.

Reflexión

EMU 8086 será el IDE que usaremos todo el semestre por ello para iniciar no esta mal explorar sus menús para ver que nos ofrecen, para así ganar agilidad y comprensión a la hora de ejecutar nuestros programas.

Actividad 3 – Parte 2

Descripción

Realice una tabla identificando de una línea de instrucciones el OPCODE, el OPERANDO y el modo de direccionamiento. Tres ejemplos de cada categoría.

Desarrollo y Resultados

Modo de Direccionamiento	Instrucción	OPCODE	OPERANDO
Modo Registro	MOV AL,BL	MOV AL	BL
	MOV DX, CX	MOV DX	CX
	ADD BH,AH	ADD BH	AH
Modo Inmediato	ADD CX, FFh	ADD CX	FFh
	MOV BL, 3Ah	MOV BL	3Ah
	MOV AX, 500	MOV AX	500
Modo Directo	MOV AX,TABLA	MOV AX	DS:TABLA
	MOV [123h], AX	MOV [123h]	AX
	MOV AL, [3]	MOV AL	[3]
Modo Registro Indirecto	MOV [BP],CX	MOV [BP]	CX
	MOV BX, [CX]	MOV BX	[CX]
	MOV AL, [SI+BL]	MOV AL	[SI+BL]

Modo Relativo a Base	ADD AH, [BH+0Ah]	ADD AH	[BH+0Ah]
	MOV [BX+4], AX	MOV [BX+4]	AX
	MOV AL, [BX]+2	MOV AL	[BX]+2
Modo Indexado a Directo	MOV AX, TABLA[DI]	MOV AX	DS:TABLA[DI]
	MOV AX,[DI+DESP]	MOV AX	[DI+DESP]
	ADD [SI+DESP],BX	ADD [SI+DESP]	BX
Modo Indexado a Base	MOV CS:desp[BX][SI],CX	MOV CS:desp[BX][SI]	CX
	MOV ARRAY[BX+SI],AX	MOV ARRAY[BX+SI]	AX
	MOV AL,arreglo1[SI]	MOV AL	arreglo1[SI]

Reflexión

Los modos de direccionamiento aun ser un tema por si solos, sin embargo, estos son necesarios para usar gran parte de las funciones del lenguaje ensamblador, por esto mismo que nosotros como estudiantes los conozcamos y comprendamos en de vital importancia.

Actividad 3 – Parte 3

Descripción

Realice una tabla con las interrupciones 21H y la 10H, enfatizando la función 09H de la 21H y las funciones 02h y 06H de la 10H**.**

Desarrollo y Resultados

Función	Código Función	Parámetros	Retorno
TERMINATE PROGRAM	Int 21/AH=00h		
READ CHARACTER FROM STANDARD INPUT, WITH ECHO	Int 21/AH=01h		AL = character read
WRITE CHARACTER TO STANDARD OUTPUT	Int 21/AH=02h	DL = character to write	AL = last character output

READ CHARACTER FROM STDAUX	Int 21/AH=03h		AL = character read
WRITE CHARACTER TO STDAUX	Int 21/AH=04h	DL = character to write	
WRITE CHARACTER TO PRINTER	Int 21/AH=05h	DL = character to print	
DIRECT CONSOLE OUTPUT	Int 21/AH=06h	DL = character (except FFh)	AL = character output
DIRECT CHARACTER INPUT, WITHOUT ECHO	Int 21/AH=07h		AL = character read from standard input
CHARACTER INPUT WITHOUT ECHO	Int 21/AH=08h		AL = character read from standard input
WRITE STRING TO STANDARD OUTPUT	Int 21/AH=09h	DS:DX -> '$'-terminated string	AL = 24h (the '$' terminating the string)
BUFFERED INPUT	Int 21/AH=0Ah	DS:DX -> buffer	Buffer filled with user input
GET STDIN STATUS	Int 21/AH=0Bh		AL = status
FLUSH BUFFER AND READ STANDARD INPUT	Int 21/AH=0Ch	AL = STDIN input function to execute after flushing buffer	As appropriate for the specified input function
DISK RESET	Int 21/AH=0Dh		CF clear
SELECT DEFAULT DRIVE	Int 21/AH=0Eh	DL = new default drive	AL = number of potentially valid drive letters

OPEN FILE USING FCB	Int 21/AH=0Fh	DS:DX -> unopened File Control Block	AL = status
CLOSE FILE USING FCB	Int 21/AH=10h	DS:DX -> File Control Block	AL = status
FIND FIRST MATCHING FILE USING FCB	Int 21/AH=11h	DS:DX -> unopened FCB, may contain '?' wildcards	AL = status
FIND NEXT MATCHING FILE USING FCB	Int 21/AH=12h	DS:DX -> unopened FCB	AL = status
DELETE FILE USING FCB	Int 21/AH=13h	DS:DX -> unopened FCB	AL = status
SEQUENTIAL READ FROM FCB FILE	Int 21/AH=14h	DS:DX -> opened FCB	AL = status
SEQUENTIAL WRITE TO FCB FILE	Int 21/AH=15h	DS:DX -> opened FCB	AL = status
CREATE OR TRUNCATE FILE USING FCB	Int 21/AH=16h	DS:DX -> unopened FCB	AL = status
RENAME FILE USING FCB	Int 21/AH=17h	DS:DX -> modified FCB	AL = status
NULL FUNCTION FOR CP/M COMPATIBILITY	Int 21/AH=18h		AL = 00h
GET CURRENT DEFAULT DRIVE	Int 21/AH=19h		AL = drive
SET DISK TRANSFER AREA ADDRESS	Int 21/AH=1Ah	DS:DX -> Disk Transfer Area	

GET ALLOCATION INFORMATION FOR DEFAULT DRIVE	Int 21/AH=1Bh		AL = sectors per cluster CX = bytes per sector DX = total number of clusters DS:BX -> media ID byte
GET ALLOCATION INFORMATION FOR SPECIFIC DRIVE	Int 21/AH=1Ch	DL = drive	AL = sectors per cluster CX = bytes per sector DX = total number of clusters DS:BX -> media ID byte
NULL FUNCTION FOR CP/M COMPATIBILITY	Int 21/AH=1Dh		AL = 00h
NULL FUNCTION FOR CP/M COMPATIBILITY	Int 21/AH=1Eh		AL = 00h
GET DRIVE PARAMETER BLOCK FOR DEFAULT DRIVE	Int 21/AH=1Fh		AL = status 00h successful DS:BX -> Drive Parameter Block
NULL FUNCTION FOR CP/M COMPATIBILITY	Int 21/AH=20h		AL = 00h
READ RANDOM RECORD FROM FCB FILE	Int 21/AH=21h	DS:DX -> opened FCB	AL = status

WRITE RANDOM RECORD TO FCB FILE	Int 21/AH=22h	DS:DX -> opened FCB	AL = status
GET FILE SIZE FOR FCB	Int 21/AH=23h	DS:DX -> unopened FCB	AL = status
SET RANDOM RECORD NUMBER FOR FCB	Int 21/AH=24h	DS:DX -> opened FCB	
SET INTERRUPT VECTOR	Int 21/AH=25h	AL = interrupt number DS:DX -> new interrupt handler	
CREATE NEW PROGRAM SEGMENT PREFIX	Int 21/AH=26h	DX = segment at which to create PSP	AL destroyed
RANDOM BLOCK READ FROM FCB FILE	Int 21/AH=27h	CX = number of records to read DS:DX -> opened FCB	AL = status CX = number of records read
RANDOM BLOCK WRITE TO FCB FILE	Int 21/AH=28h	CX = number of records to write DS:DX -> opened FCB	AL = status CX = number of records written
PARSE FILENAME INTO FCB	Int 21/AH=29h	AL = parsing options DS:SI -> filename string ES:DI -> buffer for unopened FCB	AL = result code DS:SI -> first unparsed character ES:DI buffer filled with unopened FCB
GET SYSTEM DATE	Int		CX = year

	21/AH=2Ah		DH = month DL = day AL = day of week
SET SYSTEM DATE	Int 21/AH=2Bh	CX = year DH = month DL = day	AL = status
GET SYSTEM TIME	Int 21/AH=2Ch		CH = hour CL = minute DH = second DL = 1/100 seconds
SET SYSTEM TIME	Int 21/AH=2Dh	CH = hour CL = minute DH = second DL = 1/100 seconds	AL = status
SET VERIFY FLAG	Int 21/AH=2Eh	DL = 00h AL = new state of verify flag	
GET DISK TRANSFER AREA ADDRESS	Int 21/AH=2Fh		ES:BX -> current DTA
GET DOS VERSION	Int 21/AH=30h	AL = what to return in BH	AL = major version number AH = minor version number BL:CX = 24-bit user serial number BH = MS-DOS OEM number

			BH = version flag
TERMINATE AND STAY RESIDENT	Int 21/AH=31h	AL = return code DX = number of paragraphs to keep resident	Never
GET DOS DRIVE PARAMETER BLOCK FOR SPECIFIC DRIVE	Int 21/AH=32h	DL = drive number	AL = status
EXTENDED BREAK CHECKING	Int 21/AH=33h	AL = subfunction	DL = new state
GET ADDRESS OF INDOS FLAG	Int 21/AH=34h		ES:BX -> one-byte InDOS flag
GET INTERRUPT VECTOR	Int 21/AH=35h	AL = interrupt number	ES:BX -> current interrupt handler
GET FREE DISK SPACE	Int 21/AH=36h	DL = drive number	AX = sectors per cluster BX = number of free clusters CX = bytes per sector DX = total clusters on drive
AVAILDEV - SPECIFY \DEV\ PREFIX USE	Int 21/AH=37h	AL = subfunction	DL = new state
GET COUNTRY-SPECIFIC INFORMATION	Int 21/AH=38h	AL = 00h get current-country info DS:DX -> buffer for returned info	AX = country code
MKDIR - CREATE	Int	DS:DX -> ASCIZ	CF clear if

SUBDIRECTORY	21/AH=39h	pathname	successful AX destroyed CF set on error AX = error code
RMDIR - REMOVE SUBDIRECTORY	Int 21/AH=3Ah	DS:DX -> ASCIZ pathname of directory to be removed	CF clear if successful AX destroyed CF set on error AX = error code
CHDIR - SET CURRENT DIRECTORY	Int 21/AH=3Bh	DS:DX -> ASCIZ pathname to become current directory	CF clear if successful AX destroyed CF set on error AX = error code
CREAT - CREATE OR TRUNCATE FILE	Int 21/AH=3Ch	CX = file attributes DS:DX -> ASCIZ filename	CF clear if successful AX = file handle CF set on error AX = error code
OPEN - OPEN EXISTING FILE	Int 21/AH=3Dh	AL = access and sharing modes DS:DX -> ASCIZ filename CL = attribute mask of files to look for	CF clear if successful AX = file handle CF set on error AX = error code
CLOSE - CLOSE FILE	Int 21/AH=3Eh	BX = file handle	CF clear if successful AX destroyed CF set on error

			AX = error code
READ - READ FROM FILE OR DEVICE	Int 21/AH=3Fh	BX = file handle CX = number of bytes to read DS:DX -> buffer for data	CF clear if successful AX = number of bytes actually read CF set on error AX = error code
WRITE - WRITE TO FILE OR DEVICE	Int 21/AH=40h	BX = file handle CX = number of bytes to write DS:DX -> data to write	CF clear if successful AX = number of bytes actually written CF set on error AX = error code
UNLINK - DELETE FILE	Int 21/AH=41h	DS:DX -> ASCIZ filename CL = attribute mask for deletion	CF clear if successful AX destroyed AL seems to be drive of deleted file CF set on error AX = error code
LSEEK - SET CURRENT FILE POSITION	Int 21/AH=42h	AL = origin of move BX = file handle CX:DX = (signed) offset from origin of new file position	CF clear if successful DX:AX = new file position in bytes from start of file CF set on error AX = error code
DUP - DUPLICATE FILE HANDLE	Int 21/AH=45h	BX = file handle	CF clear if successful

			AX = new handle CF set on error AX = error code
DUP2, FORCEDUP - FORCE DUPLICATE FILE HANDLE	Int 21/AH=46h	BX = file handle CX = file handle to become duplicate of first handle	CF clear if successful CF set on error AX = error code
CWD - GET CURRENT DIRECTORY	Int 21/AH=47h	DL = drive number DS:SI -> 64-byte buffer for ASCIZ pathname	CF clear if successful AX = 0100h (undocumented) CF set on error AX = error code
ALLOCATE MEMORY	Int 21/AH=48h	BX = number of paragraphs to allocate	CF clear if successful AX = segment of allocated block CF set on error AX = error code BX = size of largest available block
FREE MEMORY	Int 21/AH=49h	ES = segment of block to free	CF clear if successful CF set on error AX = error code
RESIZE MEMORY BLOCK	Int 21/AH=4Ah	BX = new size in paragraphs ES = segment of block to resize	CF clear if successful CF set on error AX = error code BX = maximum

			paragraphs available for specified memory block
LOAD AND/OR EXECUTE PROGRAM	Int 21/AH=4Bh	AL = type of load DS:DX -> ASCIZ program name ES:BX -> parameter block CX = mode	CF clear if successful BX,DX destroyed CF set on error AX = error code
EXIT - TERMINATE WITH RETURN CODE	Int 21/AH=4Ch	AL = return code	
GET RETURN CODE (ERRORLEVEL)	Int 21/AH=4Dh		AH = termination type AL = return code CF clear
FINDFIRST - FIND FIRST MATCHING FILE	Int 21/AH=4Eh	AL = special flag for use by APPEND CX = file attribute mask DS:DX -> ASCIZ file specification	CF clear if successful Disk Transfer Area filled with FindFirst data block CF set on error AX = error code
FINDNEXT - FIND NEXT MATCHING FILE	Int 21/AH=4Fh	Disk Transfer Area contains data block from previous FindFirst or FindNext call	CF clear if successful Disk Transfer Area updated CF set on error AX = error code

internal - SET CURRENT PROCESS ID (SET PSP ADDRESS)	Int 21/AH=50h	BX = segment of PSP for new process	
internal - GET CURRENT PROCESS ID (GET PSP ADDRESS)	Int 21/AH=51h		BX = segment of PSP for current process
internal - SYSVARS - GET LIST OF LISTS	Int 21/AH=52h		ES:BX -> DOS list of lists
TRANSLATE BIOS PARAMETER BLOCK TO DRIVE PARAM BLOCK	Int 21/AH=53h	DS:SI -> BIOS Parameter Block ES:BP -> buffer for Drive Parameter Block DBP drive byte must be set to valid drive	ES:BP buffer filled
GET VERIFY FLAG	Int 21/AH=54h		AL = verify flag
CREATE CHILD PSP	Int 21/AH=55h	DX = segment at which to create new PSP SI = value to place in memory size field at DX:[0002h]	AL destroyed
RENAME - RENAME FILE	Int 21/AH=56h	DS:DX -> ASCIZ filename of existing file ES:DI -> ASCIZ new filename CL = attribute mask	CF clear if successful CF set on error AX = error code
GET OR SET MEMORY ALLOCATION STRATEGY	Int 21/AH=58h	AL = subfunction	CF clear if successful CF set on error

			AX = error code
GET EXTENDED ERROR INFORMATION	Int 21/AH=59h	BX = 0000h	AX = extended error code BH = error class BL = recommended action CH = error locus ES:DI may be pointer CL, DX, SI, BP, and DS destroyed
CREATE TEMPORARY FILE	Int 21/AH=5Ah	CX = file attribute DS:DX -> ASCIZ path ending with a '\' + 13 zero bytes to receive the generated filename	CF clear if successful AX = file handle opened for read/write in compatibility mode DS:DX pathname extended with generated name for temporary file CF set on error AX = error code
CREATE NEW FILE	Int 21/AH=5Bh	CX = file attribute DS:DX -> ASCIZ filename	CF clear if successful AX = file handle opened for read/write in compatibility mode CF set on error AX = error code
FLOCK - RECORD	Int	AL = subfunction	CF clear if

LOCKING	21/AH=5Ch	BX = file handle CX:DX = start offset of region within file SI:DI = length of region in bytes	successful CF set on error AX = error code
CANONICALIZE FILENAME OR PATH	Int 21/AH=60h	DS:SI -> ASCIZ filename or path ES:DI -> 128-byte buffer for canonicalized name	CF set on error AX = error code ES:DI buffer unchanged CF clear if successful AH = 00h or 3Ah AL = destroyed
UNUSED (RESERVED FOR NETWORK USE)	Int 21/AH=61h		L = 00h
GET CURRENT PSP ADDRESS	Int 21/AH=62h		BX = segment of PSP for current process
SET VIDEO MODE	Int 10/AH=00h	AL = desired video mode	AL = video mode flag AL = CRT controller mode byte
SET TEXT-MODE CURSOR SHAPE	Int 10/AH=01h	CH = cursor start and options CL = bottom scan line containing cursor	
SET CURSOR	Int	BH = page number	

POSITION	10/AH=02h	DH = row DL = column	
GET CURSOR POSITION AND SIZE	Int 10/AH=03h	BH = page number	AX = 0000h CH = start scan line CL = end scan line DH = row DL = column
READ LIGHT PEN POSITION (except VGA)	Int 10/AH=04h		AH = light pen trigger flag DH,DL = row,column of character light pen is on CH = pixel row CX = pixel row BX = pixel column
SELECT ACTIVE DISPLAY PAGE	Int 10/AH=05h	AL = new page number	
SCROLL UP WINDOW	Int 10/AH=06h	AL = number of lines by which to scroll up BH = attribute used to write blank lines at bottom of window CH,CL = row,column of window's upper left corner DH,DL = row,column of window's lower right corner	
SCROLL DOWN WINDOW	Int 10/AH=07h	AL = number of lines by which to scroll down	

		BH = attribute used to write blank lines at top of window CH,CL = row,column of window's upper left corner DH,DL = row,column of window's lower right corner	
READ CHARACTER AND ATTRIBUTE AT CURSOR POSITION	Int 10/AH=08h	BH = page number	AH = character's attribute AH = character's color AL = character
WRITE CHARACTER AND ATTRIBUTE AT CURSOR POSITION	Int 10/AH=09h	AL = character to display BH = page number BL = attribute CX = number of times to write character	
WRITE CHARACTER ONLY AT CURSOR POSITION	Int 10/AH=0Ah	AL = character to display BH = page number background color in 256-color graphics modes BL = attribute CX = number of times to write character	
SET BACKGROUND/BORDER COLOR	Int 10/AH=0Bh	BH = 00h BL = background/border color	

WRITE GRAPHICS PIXEL	Int 10/AH=0Ch	BH = page number AL = pixel color CX = column DX = row	
READ GRAPHICS PIXEL	Int 10/AH=0Dh	BH = page number CX = column DX = row	AL = pixel color
TELETYPE OUTPUT	Int 10/AH=0Eh	AL = character to write BH = page number BL = foreground color	
GET CURRENT VIDEO MODE	Int 10/AH=0Fh		AH = number of character columns AL = display mode BH = active page
SET WINDOW COORDINATES	Int 10/AH=10h	CH,CL = row,column of upper left corner of window DH,DL = row,column of lower right corner of window	AL = status AH destroyed
GET WINDOW COORDINATES	Int 10/AH=11h		CH,CL = row,column of upper left corner DH,DL = row,column of lower right corner
GET BLANKING ATTRIBUTE	Int 10/AH=12h		BH = attribute to use on blanked lines when scrolling

WRITE STRING (AT and later,EGA)	Int 10/AH=13h	AL = write mode CX = number of characters in string. DH,DL = row,column at which to start writing. ES:BP -> string to write	
GET PHYSICAL DISPLAY PARAMETERS (CONVERTIBLE)	Int 10/AH=15h		AX = alternate display adapter type ES:DI -> parameter table
GET/SET FONT PATTERN	Int 10/AH=18h	AL = subfunction BX = 0000h CL = character size in bytes (01h,02h) CH = 00h DH = character width in pixels DL = character height in pixels ES:DI -> buffer for/containing font image	AL = status ES:DI buffer filled for function 00h if successful

http://www.ctyme.com/intr/int.htm

Reflexión

Atrás ver la interminable lista de dos de las 255 diferentes interrupciones que existen en el leguaje ensamblador del 8086 llegue a la conclusión que es completamente imposible llegar a memorizar o conocerlas todas, sin embargo, actividades como esta nos ayuda a darnos una idea general de algunas de las funciones que nos ofrecen estas dos interrupciones.