UNIVERSIDAD PEDAGÓGICA Y TECNOLÓGICA DE COLOMBIA
FACULTAD SEDE SOGAMOSO
ESCUELA DE INGENIERÍA ELECTRÓNICA
LABORATORIO DE MICROPROCESADORES

PRÁCTICA Nº 1

USO DEL SET DE INSTRUCCIONES INTEL X86 EN EMU8086

I. OBJETIVOS

- Conocer el entorno para el desarrollo de aplicaciones en ensamblador para Intelx86 denominadoEmu8086.
- Identificarlas características de funcionamiento de las instrucciones lógicas y aritméticas para Intelx86.
- Realizar un programa en ensamblador aplicando las instrucciones lógicas y aritméticas para Intel x86, sobre el emulador Emu8086.

II. INTRODUCCIÓN

Los procesadores Intel de la familia Intel x86 disponen de una amplia variedad de instrucciones que corresponden a un repertorio tipo CISC (*ComplexInstruction Set Computer*). En el estudio de los procesadores, es fundamental conocer los diferentes tipos de instrucciones, su forma de uso y los registros y banderas que se afectan durante su ejecución.

En esta guía se abordan 5grupos de instrucciones: las instruccionespara transferencia de datos, instrucciones lógicas y aritméticas, instrucciones para la manipulación de bits, instrucciones para la transferencia de cadenas de datos e instrucciones para el control de programas.

La familia de procesadores Intel x86 pueden sumar restar multiplicar y dividir datos tales como bytes palabras y dobles palabras con signo o sin signo.Además,se pueden modificar datos al nivel de bits mediante las operaciones lógicas.

III. ELEMENTOS Y EQUIPOS NECESARIOS

- Software *Emu8086*
- Computador Personal

IV. DESCRIPCIÓN DEL LABORATORIO

En esta guía se presentan algunos programas de ejemplo que deben ser compilados y simulados en el emulador Emu8086, para conocer los aspectos básicos relacionados con el uso y funcionamiento de las instrucciones de la familia Intel x86.

Por otra parte, se propone el desarrollo de algunos programas de aplicación en los que se utilicen las instrucciones del set de instrucciones estudiado, empleando instrucciones de cada uno de los 5 grupos mencionados previamente.

Para información detallada de cada una de las instrucciones, consulte la documentación y tutoriales disponibles en línea, además de referencias bibliográficas como el libro "Los Microprocesadores Intel" de Barry Brey 8ª Ed y los manuales de los microprocesadores Intel.Asimismo, el emulador Emu8086 dispone de información relativa al set de instrucciones y el conjunto de interrupciones soportado, la cual puede ser accedida dando click en el botón "**help**" del menú de herramientas, ver Fig. 1.

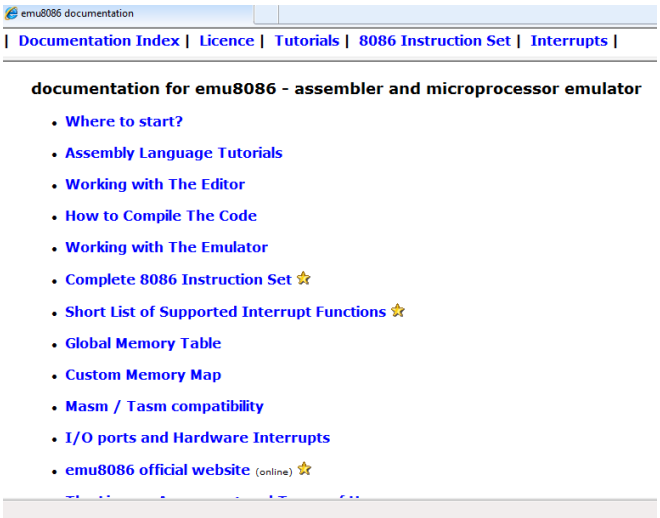


Figura 1. Documentación y tutoriales del Emu8086

En las pestañas "***Tutorials"*** y "***8086 Instruction Set"*** encontrará la información necesaria referente al set de instrucciones.

a) Creación de un nuevo proyecto en Emu8086

Para la creación de un nuevo proyecto en Emu8086 es necesario abrir el programa y seleccionar nuevo (New) como se muestra en la Figura 2.

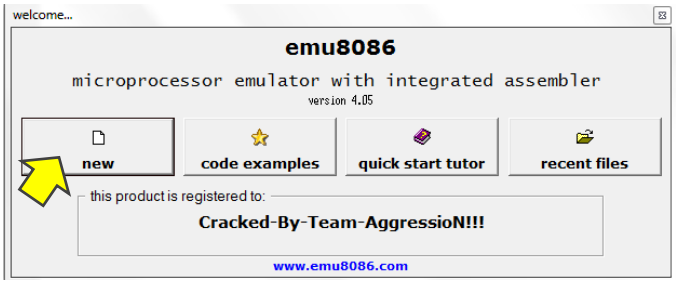


Figura 2. Opciones de inicio para un nuevo proyecto.

A continuación aparece una ventana en la que se puedenseleccionar plantillas preestablecidas según el tipo de archivoa emular, tales como archivos ***.COM***, ***.EXE***,entre otros. Para estosprimerosejerciciosmarcamosla opción espacio de trabajo en blanco (emptyworkspace), ya que no deseamos emplear ninguna platilla, ver la Figura 3.

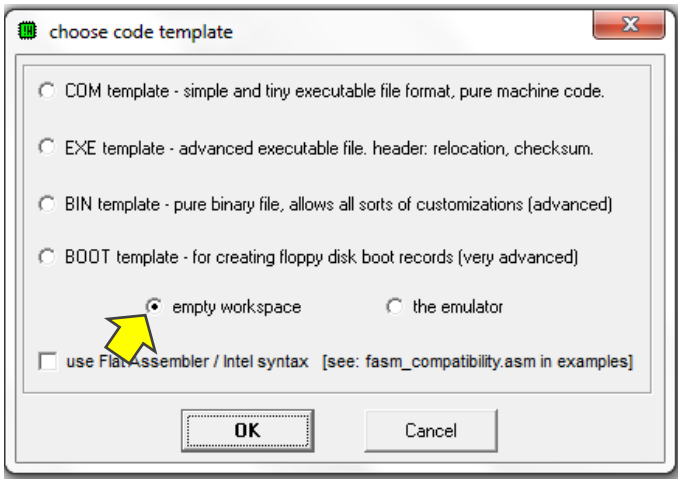


Figura 3. Selección del tipo de archivo o plantilla para trabajar.

Ahora vamos a construir un sencillo ejemplo de un programa en ensamblador. El objetivo del programa es desplegar un mensaje en pantalla mediante el uso de la opción 09 de la interrupción 21H. A continuación se muestra el código requerido:

```
ORG 100H
        JMP START
        MENSAJE DB 'MICROPROCESADORES', '$'

                ; Inicializar DS
        START:  MOV AX, @DATA
                MOV DS, AX

                ; Desplegar mensaje
                MOV DX, OFFSET MENSAJE
                MOV AH, 09H
                INT 21H

                ; Retorno al Sistema Operativo
                MOV AX, 4C00H
                INT 21H
        END START
END
```

Descripción del Código

Este ejemplo corresponde a un programa .COM, que emplea un solo segmento, y tiene reservados los primeros 256 bytes del Segmento de Código (PSP), por lo que las instrucciones empiezan a codificarse a partir de la posición 0100H (o 256)dentro del segmento. Esta ubicación del inicio del programa se consigue con la directiva ORG 100H. Luego aparece la instrucción JMP START que produce un salto de programa desde la posición 0100H a la dirección correspondiente a la etiqueta START. Con esto se consigue que exista un espacio de memoria destinado a contener los datos requeridos en el programa (la tabla con el mensaje a desplegar en este caso).

A continuación aparece el cuerpo del programa en el que se empieza por establecer el Segmento de Datos asociado con el Segmento de Código actual. Esto se logra mediante las instrucciones

```
MOV AX, @DATA
MOV DS, AX
```

En este caso, debido a que estamos creando un archivo .COM, el resultado de la ejecución de estas instrucciones producirá que el registro DS contenga el mismo valor del registro CS.

Luego se encuentra un grupo de tres instrucciones dedicadas a desplegar un mensaje en pantalla (en forma de un vector de caracteres ASCII) mediante el uso de la función 09H de la interrupción 21H. Para esto se establece el desplazamiento de la etiqueta MENSAJE. Este desplazamiento se guarda en el registro DX que trabaja como registro de dirección que apunta al inicio del vector de caracteres ASCII. A continuación se invoca la función 09H de la interrupción 21H. Esta función trabaja de forma automática desplegando cada uno de los caracteres en pantalla mientras no aparezca el caracter '$', que actúa como caracter de finalización.

Finalmente se cierra la ejecución del programa mediante la función 4CH de la interrupción 21H, devolviendo el control al sistema operativo. El registro AL contiene 00H como código de retorno de error. Se cierra el procedimiento mediante la directiva END START y se cierra el programa mediante la directiva END.

Este mismo programa se puede construir de forma más compacta, empleando el estilo de modelos de memoria. Los tipos de modelos de memoria que se pueden emplear son .TINY, .SMALL, .MEDIUM, .COMPACT, .LARGE,.HUGE, y .FLAT.

En este caso se deben definir los segmentos a emplear en el programa, correspondientes a los segmentos de datos (.DATA), código (.CODE) y pila (.STACK).Los únicos dos segmentos que se deben declarar'obligatoriamente' son el segmento de código y el de pila. A continuación se muestra el mismo programa presentado anteriormente, pero descrito esta vez por modelos de memoria.

```
.MODEL SMALL
.STACK 200
.DATA
        MENSAJE DB 'MICROPROCESADORES', '$'
.CODE


        ; Inicializar DS
        START:  MOV AX, @DATA
                MOV DS, AX

                ; Desplegar mensaje
                MOV DX, OFFSET MENSAJE
                MOV AH, 09H
                INT 21H

                ; Retorno al Sistema Operativo
                MOV AX, 4C00H
                INT 21H
        END START
END
```

b) *Definición de procedimientos*

Un procedimiento (o subrutina) es un grupo de instrucciones que desempeñan una tarea o rutina específica. La definición de un procedimiento se realiza de la siguiente manera:

```
NOMBRE PROC NEAR
        En este espacio se ingresan las instrucciones de la subrutina
        RET
        ENDP
```

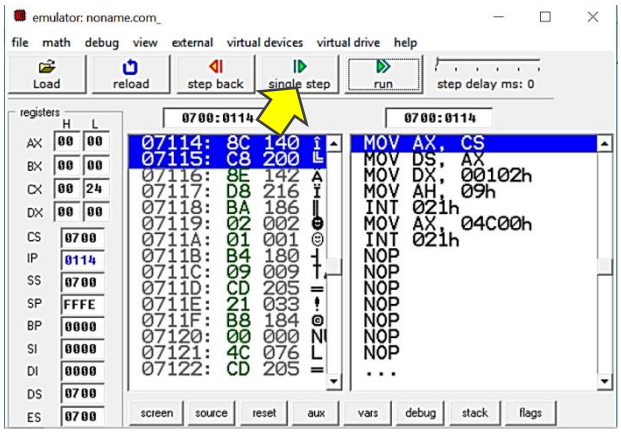
La directiva PROC indica que se está declarando un nuevo procedimiento, mientras que la directiva NEAR indica que éste procedimiento se declara para ser llamado de forma INTRASEGMENTO, es decir, teniendo en cuenta solo el desplazamiento de la etiqueta NOMBRE (la etiqueta que corresponda al nombre del procedimiento). Se puede emplear también la directiva FAR, que indicaría que el procedimiento ha sido diseñado para ser llamado de forma EXTRASEGMENTO, es decir, teniendo en cuenta tanto el desplazamiento como el segmento donde éste ha sido declarado. En tal caso el segmento de código que contenga al procedimiento deberá ser declarado como de tipo 'público'.Para invocar y ejecutar el procedimiento creado basta con ejecutarla instrucción

CALL NOMBRE.

c) *Compilación y emulación de programas*

Para cada uno de los problemas propuestos, proceda como se indica a continuación:

1. Abra el emulador de Intel x86 'Emu8086' y digite el programa respectivo.
2. Compile y emule programa dandoclick en el icono (emulate) de la barra de herramientas el Emu8086. Ejecute el programa paso a paso dando click en el botón 'single step'. Observe detalladamente el estado de cada una de las banderas y registros que se ven afectados durante la ejecución del programa.



3. El trabajo será evaluado al finalizar la clase. Recuerde que el código se debe presentar identado y comentado, de manera que sea fácil de seguir y entender.

d) Ejercicio base

En esta parte se identificaran las instrucciones aritméticas del x86 basado en un programa que se presenta de la siguiente forma aritmética:

$$(25_{10}+15_{10})(625_{10}-250_{10})+191_{10}$$

Abriendo un nuevo proyecto en el simulador EMU8086 y usando el editor de ensamblador, ingresar el siguiente código:

```
ORG 100H
MOV AX, 27    ; Move 27₁₀ to AX
ADD AX, 15    ; Add 15₁₀ to AX, store result in AX
MOV BX, 625   ; Move 625₁₀ to BX
SUB BX, 250   ; Subtract 250₁₀ from BX, store result in BX
MUL BX        ; Multiply BX to AX, store result in AX (and DX)
ADD AX, 191   ; Add 191₁₀ to AX, store result in AX
RET
```

Iniciar la simulación y verificar paso a paso el funcionamiento del código.

e) Ejercicios propuestos

1. Diseñe un programa en ensamblador que calcule el resultado de una operación matemática básica haciendo uso de las instrucciones aritméticas. Dicha operación debe tener como mínimo seis operandos y debe ser ingresada por teclado y su resultado debe ser visualizado por pantalla.
2. Diseñe un programa en ensamblador que calcule la suma y la multiplicación de dos matrices 4 x 4. Los datos de entrada deben ser dispuestos en dos arreglos de memoria, así como las matrices de suma y multiplicación. Presente las matrices de entrada y los resultados en pantalla.
3. Desarrolle un programa en ensamblador Evaluar el determinante de matrices 3X3. Presente las matrices de entrada y los resultados en pantalla.

BIBLIOGRAFÍA

[1] Barry B. Brey. The Intel microprocessors 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, Pentium Pro processor, Pentium II, Pentium III, Pentium 4, and Core2 with 64-bit extensions: architecture, programming, and interfacing. 8th ed.2009.

[2] ABEL, Peter. Lenguaje ensamblador y programación para IBM PC y compatibles. Pearson Educación, 1996.

[3] GODFREY, J. Ferry; TERRY, J. Lenguaje ensamblador para microcomputadoras IBM. Editorial Prentice-Hall, 1991.

[4] Ensamblador 8086/88. Universidad de Sevilla. http://www.cartagena99.com/recursos/programacion/apuntes/Esamblador8086.pdf

[5] Emu8086 Documentation. http://www.ee.hacettepe.edu.tr/~alkar/ELE336/emu8086.pdf