Guía para el desarrollo del TP N°5
PROGRAMACIÓN DE SIMULPROC
(Procesador virtual)

Introducción

Como ya se explicó en la guía teórica, la programación de computadoras debe hacerse en un lenguaje que sea comprensible por las mismas (lenguaje de máquina) o en lenguajes que puedan finalmente ser traducidos al lenguaje que las computadoras pueden interpretar.

Por ello en esta práctica vamos a investigar como es el proceso de ejecución y programación en el primer nivel de lenguaje que le sigue al lenguaje de máquina, **el lenguaje ensamblador**. En este camino nos vamos a valer de un programa de simulación de un procesador virtual llamado SimuProc.

El objetivo estará en recorrer desde el principio el camino para desarrollar un programa de computadora. En este proceso se verá como se pasa por cada nivel de abstracción hasta llegar a la perspectiva del hardware presentada por el simulador que vamos a estudiar

El autor de Simuproc es Vladimir Yepes y autoriza su libre distribución.

Descripción del simulador SimuProc

SimuProc es un simulador de un Procesador Hipotético con el cual podrás aprender las nociones básicas para empezar a programar en lenguaje ensamblador, en el cual podemos observar todo el proceso interno de ejecución del programa a través de cada ciclo del procesador.

Este Simulador Hipotético muestra como funciona un procesador internamente. En el cual se puede ver que realiza este en cada Ciclo.
El Ciclo de ejecución de un Procesador se divide en 2 semi-ciclos simples:

1. **El ciclo de Fetch**
   -Va al PC
   -Va a la dirección que apunta el PC
   -Hace IR = MEM[PC]
   -Incrementa PC
2. **El ciclo de Ejecución**
   -Si tiene que ir a Memoria
   -va a Memoria
   -ejecuta instrucción
   -almacena resultados.

Características del procesador.

Memoria:
La Memoria es el dispositivo que almacena toda la información del programa que se ejecuta, tanto datos como instrucciones.
Esta en realidad no es parte del procesador, sino que es un dispositivo aparte al que el procesador accede para ir leyendo las instrucciones y datos del programa. La capacidad de la memoria Simulada es de 4096 posiciones de 16 bits cada registro:
Desde 000 hasta FFF.
El simulador trabaja con constantes y variables en binario y direcciones (posiciones de memoria) en Hexadecimal.

Registros Generales:
Los registros generales del procesador se usan para almacenar información de uso rápido, ya que se accede a ellos a una velocidad mucho más alta que la memoria. En ellos se pueden almacenar direcciones de memoria a las que se va a acceder bastante a lo largo de la ejecución del programa, o directamente variables que se desean usar.
Este Procesador consta de 3 registros de propósito general, AX, BX y CX cada uno con 16 bits de capacidad.

Registros Apuntadores como:
PC IP: Program Counter o Instruction Pointer, Contiene la dirección de memoria de la próxima instrucción a ejecutar y es incrementado en cada nueva instrucción.

MAR: Memory Address Register. (Registro de Dirección de Memoria) es el registro en el que se almacena la dirección de memoria a la que se quiere acceder.

MDR: Memory Data Register o Memory Buffer Register, es un registro intermedio en el que se almacenan los datos que se escriben o leen de memoria. En el caso de una lectura, se pone en el MAR la dirección y se activa la señal de leer, obteniendo en el MDR el dato buscado. En el caso de una escritura, se pone en el MAR la dirección y en el MDR el dato a escribir en memoria, después de activa la señal de escribir, de esta forma almacenamos en memoria el dato.

IR: Instruction Register, en este registro se introduce la instrucción a ejecutar, después de haberla leído de la memoria accediendo a ella mediante la dirección señalada en el PC; El contenido de este registro se puede dividir en código de operación (el código que señala la operación que se realizará) y el o los operandos. Puede haber 2 operandos o uno solo. Acá es donde se decodifica e interpreta la instrucción así: se descompone la instrucción leída de forma que se pueda saber cual es la operación que se desea realizar y cuales son los operandos, en su caso, o el desplazamiento en caso de que se trate de una instrucción de bifurcación...

Registros de Pila:
BP: Base Pointer, Puntero de base de la pila. El valor de por defecto es F80, Este puede cambiarse desde un programa, asignándole otra dirección de memoria con la instrucción MOV. Digamos que quiero reservar mas espacio para la pila haciendo que esta comience desde la posición CF1, entonces copio esta dirección de memoria en cualquier posición de memoria; supongamos que lo copie en la dirección 3B entonces uso la instrucción MOV BP, 3B y así BP es igual a CF1. Mientras se ejecuta el programa se puede visualizar en una barra de porcentaje el uso de la pila.

SP: Stack Pointer, Puntero de la pila, indica en que próxima dirección de la pila esta disponible, es decir, apunta a la cima de la pila. Este valor se cambia automáticamente cuando se usan las instrucciones PUSH POP.

Registros de Control (Flags) o Registros de estado
Estos registros se usan para poder controlar el comportamiento de un programa los cuales se activan después de cada operación, según sea el resultado de la instrucción ejecutada.
Zero flag: se vuelve 1 si el resultado de la ultima operación = 0
Negative Sign flag: Se vuelve 1 si el resultado de la última operación es igual a un número negativo.
Carry flag: se activa cuando la operación realizada ha producido un acarreo.
Overflow flag: se activa cuando la operación produjo desbordamiento (overflow), es decir, el resultado ocupaba más de los 16 bits que caben en un registro.

Estos flags se usan principalmente en instrucciones de *bifurcación* (por ejemplo, si queremos que, en caso de que el resultado de la última operación fuera cero, el programa se salte varias de las instrucciones siguientes, comprobamos el flag cero y si está activo el programa salta, esto se consigue con la instrucción JEQ).

En la **ALU** (Arithmetic Logic Unit) - (Unidad Aritmética y Lógica) es donde el procesador realiza las operaciones matemáticas, (suma, resta, multiplicación...) y las operaciones lógicas (AND, OR, desplazamiento de bits...).

Descripción de la interfaz gráfica del programa

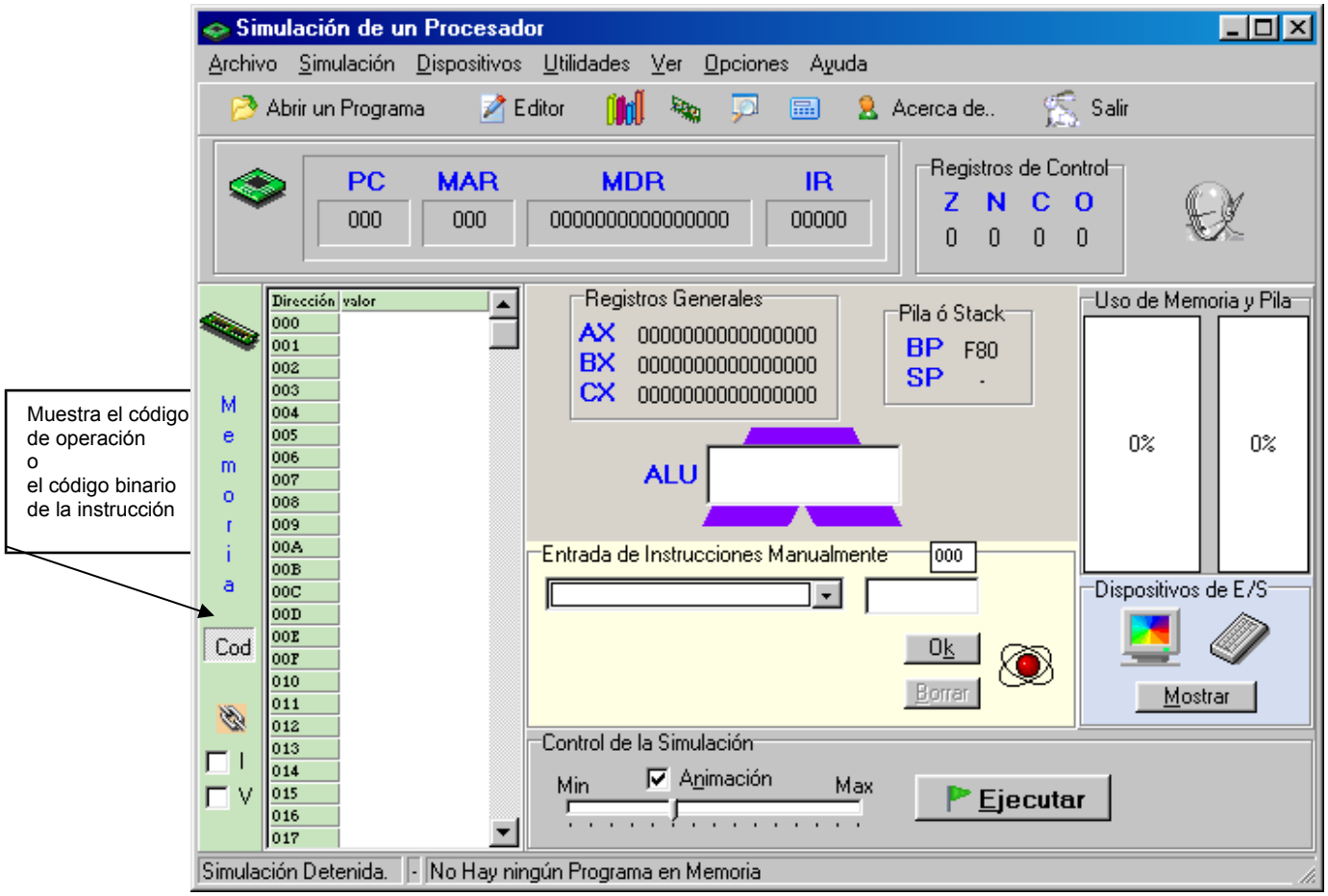

figura-A

En la **figura-A** podemos observar como se representan y distribuyen en la ventana de ejecución los componentes del procesador descriptos anteriormente.

Control de simulación
En la figura-A también se observa un control de velocidad de simulación que nos permite graduar la rapidez con la que el procesador ejecutará el programa cargado en memoria.

Ventana de entrada/salida de datos
Se acciona picando sobre el botón "Mostrar". Aquí podremos interactuar con el procesador observando los resultados que va arrojando el programa en ejecución o bien ingresando datos requeridos por el programa

Definiciones generales
Por medio de los controles que se mencionaron esta interfaz gráfica nos permitirá observar e intervenir en la ejecución del programa, donde veremos como se carga una instrucción de memoria en los registros del procesador, como se decodifica y realizan cálculos en la ALU, como se modifican los registros de estado según el resultado de las operaciones matemáticas realizadas en la ALU

LAS INSTRUCCIONES MATEMATICAS PUEDEN MODIFICAR LOS BITS DEL REGISTRO DE ESTADO

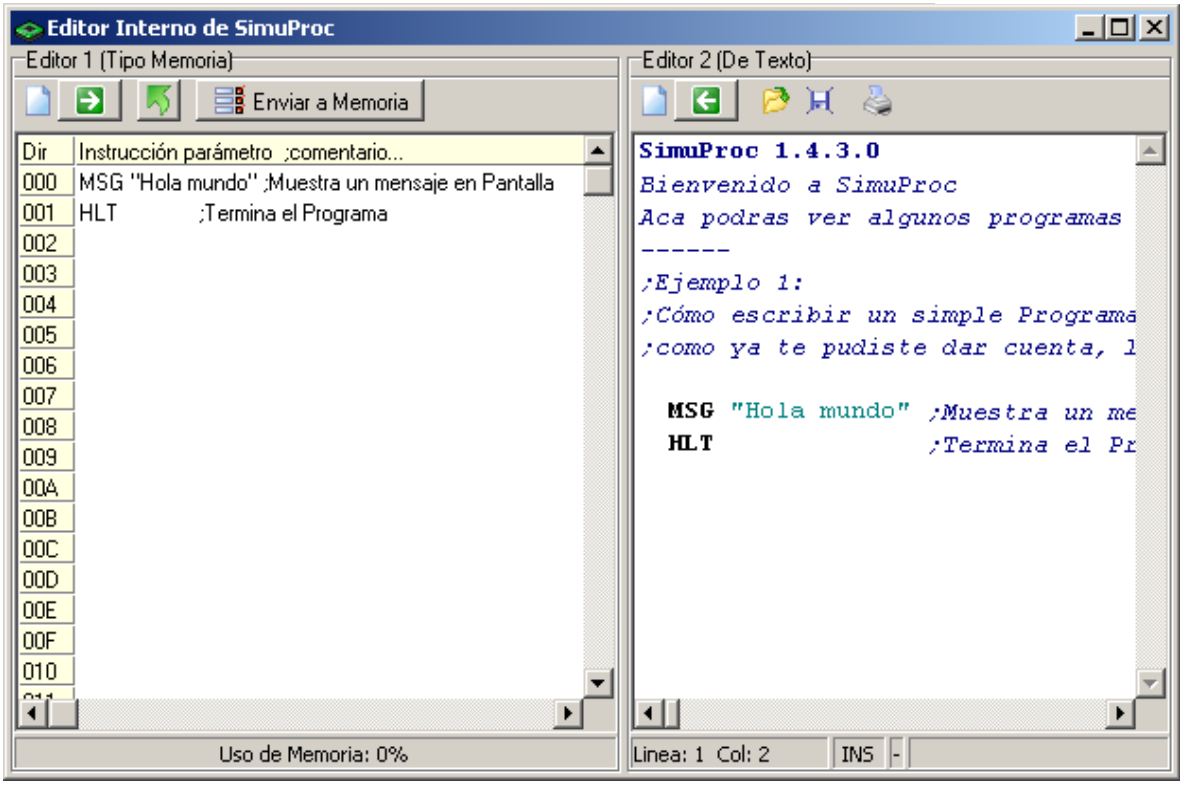

figura-B

En la figura-B observamos la estructura del editor de programas.

Editor 2 (Ventana derecha)
Aquí es donde comenzaremos con el proceso de implementación de los algoritmos obtenidos en la fase de resolución. Para este fin comenzaremos introduciendo los mnemónicos de las instrucciones que vamos a ejecutar. También podemos agregar comentarios que hagan más sencilla la lectura e interpretación del algoritmo codificado en lenguaje ensamblador.

La documentación en muy importante en las fases de prueba y depuración, simuproc nos permite agregar comentarios para este fin

Editor 1 (Ventana izquierda)
En esta editor veremos como quedarán distribuidas en la memoria las instrucciones ingresadas en el editor 1. También tiene un asistente de corrección de errores donde acusará con un mensaje todas aquellas inconsistencias que pudieran tener las instrucciones cargadas.

Especificaciones de Simulproc

Memoria RAM:

- 4096 posiciones (o 512 Bytes) de 16bits c/palabra de mem
- Primer posición = 000x0
- Última posición = FFFx0
- Capacidad en Bytes (8bits) = 8KBytes (posc de mem x 2bytes per word)

Registros (AX,BC,CX,MAR,MDR,IR):

- 3 registros de propósito gral. de 16bits c/u (AX, BX, CX)
- 1 *Registro apuntador* PC o IP (program counter o instruction pointer) almac **prox**. Dir de mem a acceder 12bits
- 1 Memory addres register almc la dir de memoria a la que **se va a acceder** 12bits
- 1 *Memory data register* o memory buffer reg, almc las palabras que se leen o escriben **de memoria** 16bits
- Máximo entero positivo aceptado por la entrada de datos: **65535**
- *Registro de instrucción*, este registro contiene el código de operación y los operandos. Aquí se decodifica e interpreta la instrucción

Registros de pila (BP,SP,STACK):

- *Puntero base* (BP), por default vale F80x0. Este puntero indica desde que dir de memoria RAM está reservado para la estructura de almac llamada **"pila"**. Como usa la memoria como medio físico hereda la longitud de palabra de 16bits de la misma.
- *Stack Pointer* o puntero de pila (SP), indica cual es la proxima dirección de la pila disponible, en otras palabras apunta a la cima de al pila.
- *Pila*, utiliza parte de la memoria del procesador como medio de almacenamiento, tiene 128 posiciones (o 16Bytes). La estrategia de acceso es FILO (first in last out)

Registro de control (Flags: Z,N,C,O):

- O también llamado *registro de estado*, permite monitorear el comportamiento de un programa en tiempo de ejecución. Esto se lleva a cabo observando los flags que se actualizan luego de la ejecución de una instrucción en el procesador.

ALU:

- *Unidad aritmética lógica*, lugar donde se realizan las operaciones matemáticas y lógicas (+,-,/,* o AND, OR, bits shift)

Funcionamiento del simulador paso a paso

Como se explica en la guía del TP5 el funcionamiento del CPU simulado por el software "SimuProc" consta de dos etapas, en la primera el CPU se ocupa de recuperar la instrucción de memoria **"FETCH"** (y obtener los datos si así la instrucción lo indica) y en la segunda de ejecutar dicha instrucción **"EXECUTE"** que puede involucrar la operación de los datos previamente recogidos en la etapa de búsqueda.
A continuación se desarrolla paso a paso las dos etapas señalando los registros usados y tareas realizadas:

1. Cargar el PC con la dirección próxima dirección de memoria 000x0 a ejecutar
2. Envía al MAR la dirección de memoria a leer
3. Se carga en el MDR el contenido de la dirección apuntada por el MAR
4. Se entrega al IR el contenido para que lo decodifique e incremente PC
5. **SI** la instrucción indica un operando se carga en el MAR la dirección de dicho dato y se continúa desde el paso 2, **DE OTRA FORMA** se ejecuta la instrucción decodificada

Descripción del formato de instrucción

Las instrucciones cuentan con dos partes, una es el código de operación y el otro es el operando.
Las instrucciones pueden utilizar dos, uno o ningún operando según el tipo de operación que sea.
Otras instrucciones tienen el operando implícito, por ejemplo CLA que pone a cero el registro AX o XAB que intercambia el contenido de los registros AX y BX entre sí.

- **Código de operación**: se expresa con un mnemónico y es representado por 2 dígitos decimales que tienen como rango de 01 a 99.
- **Operando**: puede ser un registro o una dirección de memoria

Instrucción genérica

XX - INST [op1][op2]

Donde:

- **XX** es el código de la Instrucción expresado en 2 dígitos decimales
- **INST** expresada como mnemónico
- **[op1]** es el parámetro explicito que puede indicar un registro o una dirección de memoria
- **[op2]** es el parámetro explicito que puede indicar un registro o una dirección de memoria