LABORATORIO
Arquitectura 8086

# LABORATORIO DE ARQUITECTURA DE COMPUTADORAS

Curso de Arquitectura de computadoras
Instituto de Computación
Facultad de Ingeniería

## Objetivos

El presente trabajo obligatorio tiene como objetivo introducir al estudiante al conocimiento práctico de la arquitectura 8086.
El laboratorio propone un acercamiento a la arquitectura 8086 desde su lenguaje ensamblador, incluyendo la compilación manual desde C de estructuras de datos y control y la interacción con el sistema de entrada y salida (E/S).
Finalmente, en el laboratorio se ejercitará la implementación, compilación y ejecución de rutinas recursivas.

## Descripción de la tarea

En este laboratorio se desarrollará un programa Manejador y Ordenador de Árboles de Búsqueda (MOAB). El programa permitirá administrar un Árbol Binario de Búsqueda (ABB) y realizar operaciones sobre él. Dentro de las funcionalidades ofrecidas se encuentra agregar nuevos números al árbol, imprimir información sobre él y permitir varios formatos de almacenamiento interno. Además, el sistema manejará una bitácora de ejecución, donde se indicará información sobre parámetros leídos y acciones realizadas.

## Formato de entrada

La entrada al sistema se realizará leyendo el puerto de entrada/salida de 16 bits de solo lectura **PUERTO_ENTRADA** con el formato ***Comando [Parámetro]***. Cada comando tiene predefinidos si requiere cero o un parámetro. Tanto los comandos como los parámetros son de 16 bits.

La siguiente tabla muestra la codificación requerida para cada comando:

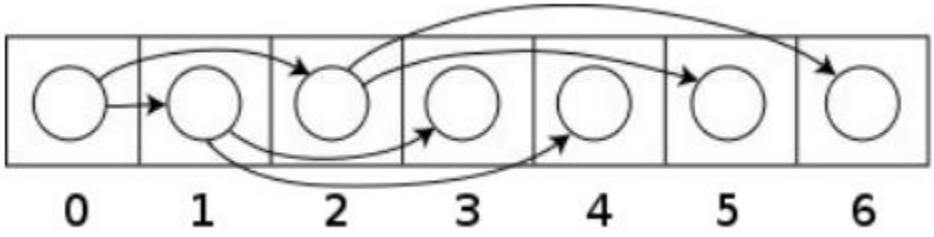
| **Comando** | **Parámetro** | **Código** | **Descripción** |
|---|---|---|---|
| Cambiar Modo | Modo | 1 | Cambia el modo de almacenamiento del árbol e inicializa el área de memoria. |
| Agregar Nodo | Número | 2 | Agrega el número al árbol. El número es un número de 16 bits en complemento a 2. |
| Calcular Altura | | 3 | Imprime la altura del árbol. |
| Calcular Suma | | 4 | Imprime la suma de todos los números del árbol. |
| Imprimir Árbol | Orden | 5 | Imprime todos los **números** del árbol: el parámetro orden indica si se imprimen de menor a mayor (0) o de mayor a menor (1) |
| Imprimir Memoria | N | 6 | Imprime los primeros N nodos del área de memoria del árbol. |
| Detener programa | | 255 | Detiene la ejecución |

## Almacenamiento del árbol

Para el almacenamiento del árbol se reserva una zona de memoria de tamaño **AREA_DE_MEMORIA** palabras de 16 bits ubicada a partir de la posición 0 del segmento ES según el formato indicado por el modo. Al cambiar el modo del árbol, se deberá inicializar el área de memoria con el valor 0x8000 en cada una de las palabras del área de memoria. Como consecuencia de esto, un cambio de modo **elimina** todos los valores previos almacenados en el árbol.

**Modo estático**

En este modo, los nodos tienen una ubicación prefijada en la memoria a partir del nodo inicial, según la siguiente figura:

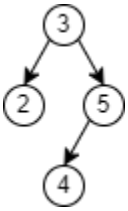



Dada esta representación, cada nodo simplemente guarda el número almacenado pues la posición del siguiente nodo es implícita. El valor 0x8000 indica que el nodo no está siendo utilizado.

El primer número agregado siempre se ubica en la posición 0 del árbol, pero los siguientes números se insertan siguiendo el patrón de almacenamiento presentado. Si los comandos recibidos son:

num 3
num 2
num 5
num 4

El árbol representado es el siguiente:




Mientras que la memoria se verá como a continuación:

| Dirección | Descripción | Valor |
|---|---|---|
| ES:0 | Nodo 0 -> num | 3 |
| ES:2 | Nodo 1 -> num | 2 |
| ES:4 | Nodo 2 -> num | 5 |
| ES:6 | Nodo 3 -> num | 0x8000 |
| ES:8 | Nodo 4 -> num | 0x8000 |
| ES:10 | Nodo 5 -> num | 4 |
| ES:12 | Nodo 6 -> num | 0x8000 |
| ES:14 | Nodo 7 -> num | 0x8000 |
| ES:16 | Nodo 8 -> num | 0x8000 |

**Modo dinámico**

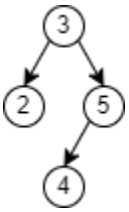
En el modo dinámico los nodos tendrán posiciones de memoria dinámicas, las cuales quedarán definidas según el orden en que se agreguen los nodos al árbol. En este modo, la estructura de cada nodo será la siguiente:

```
typedef struct{
     short num;
     short izq;
     short der;
} nodo;
```

El árbol se representará como un arreglo de nodos, donde los nodos se almacenarán uno a continuación del otro, en orden de creación (arreglo con tope). Dentro de cada nodo, izq y der son los índices dentro del arreglo árbol donde se encuentran los nodos izquierdo y derecho. Nuevamente, un valor 0x8000 en alguno de los índices izq o der indica que el nodo no tiene hijo izquierdo o derecho respectivamente. De este modo, si los comandos recibidos son:

num 3
num 2
num 5
num 4

El árbol representado es el siguiente:




Mientras que la memoria se verá como a continuación:

| **Dirección** | **Descripción** | **Valor** |
|---|---|---|
| ES:0 | Nodo 0 -> num | 3 |
| ES:2 | Nodo 0 -> izq | 1 |
| ES:4 | Nodo 0 -> der | 2 |
| ES:6 | Nodo 1 -> num | 2 |
| ES:8 | Nodo 1 -> izq | 0x8000 |
| ES:10 | Nodo 1 -> der | 0x8000 |
| ES:12 | Nodo 2 -> num | 5 |
| ES:14 | Nodo 2 -> izq | 3 |
| ES:16 | Nodo 2 -> der | 0x8000 |
| ES:18 | Nodo 3 -> num | 4 |
| ES:20 | Nodo 3 -> izq | 0x8000 |
| ES:22 | Nodo 3 -> der | 0x8000 |

## Especificación de los comandos

### Cambiar Modo

Indica el modo de almacenamiento del árbol. Si el parámetro es 0, el almacenamiento será estático, mientras que si el parámetro es 1, será dinámico. Despliega error en caso de recibir un parámetro inválido. Además, inicializa el área de memoria.

### Agregar Nodo

Agrega el parámetro Num al árbol. Imprime error en el **PUERTO_LOG** en caso de intentar escribir fuera del **AREA_DE_MEMORIA.** Imprime error en el **PUERTO_LOG** en caso de que el nodo ya esté contenido en el árbol.

### Calcular Altura

Imprime la altura del árbol en el puerto de entrada/salida **PUERTO_SALIDA**.

### Calcular Suma

Imprime la suma de todos los valores del árbol en el puerto de entrada/salida **PUERTO_SALIDA**.

### Imprimir Árbol

Imprime todos los números del árbol en el **PUERTO_SALIDA**: el parámetro *orden* indica si se imprimen de menor a mayor (0) o de mayor a menor (1).

### Imprimir Memoria

Imprime el contenido de memoria de los primeros N nodos (N es parámetro) en el **PUERTO_SALIDA**. Tener en cuenta que la cantidad de *bytes* impresos difiere según el modo de almacenamiento utilizado. En el modo estático se imprimirán 2 * N bytes (ya que cada nodo simplemente guarda los 16 bits del número), mientras que en el modo dinámico se imprimirán 6 * N bytes.

### Detener Programa

Detiene la ejecución del programa.

### Observaciones

- En caso de recibir un comando *modo* que vuelva a setear el mismo modo previamente almacenado, el comando se deberá procesar normalmente, eliminando los contenidos previos del árbol.
- Se pueden utilizar variables adicionales para el mantenimiento y almacenamiento del árbol, pero los nodos se deben almacenar según lo indicado en la letra.
- Los números agregados al árbol se encuentran representados en complemento a 2. El número 0x8000 no es válido ya que este código se utiliza para representar la ausencia de nodo.

## Bitácora de ejecución

El programa MOAB debe mantener una bitácora de ejecución a medida que va procesando cada comando. Se utilizará el puerto de salida **PUERTO_LOG**. La bitácora deberá funcionar de la siguiente forma:

- Antes de procesar un comando se debe mandar el código 64 seguido del comando a procesar (incluyendo los parámetros, una palabra por cada dato)
- Luego de procesar el comando se deberá mandar:
  - El código 0 si la operación se pudo realizar con éxito
  - El código 1 si no se reconoce el comando (comando inválido)
  - El código 2 si el valor de algún parámetro recibido es inválido.
  - El código 4 si al agregar un nodo se intenta escribir fuera del área de memoria.
  - El código 8 si el nodo a agregar ya se encuentra en el árbol.

## Puertos de entrada y salida y constantes

PUERTO_ENTRADA: 20
PUERTO_SALIDA: 21
PUERTO_LOG:  22
AREA_MEMORIA: 2048 (palabras de 16 bits)

## Herramientas

Se utiliza un ambiente simulado como plataforma de programación con el objetivo de minimizar las dependencias con el sistema operativo y el sistema computador, y fomentar la comprensión de los elementos y mecanismos de hardware de la arquitectura 8086. En 2014 la Facultad de Ingeniería se propone el desarrollo de un simulador como herramienta educativa para la enseñanza de la arquitectura 8086, iniciando así el desarrollo del simulador ArquiSim [1]. El simulador (ArquiSim) permite el desarrollo de software en ensamblador, el ensamblado del mismo y la simulación de la ejecución sobre una arquitectura 8086. Para el desarrollo de software el simulador propone estructurar el código en cuatro tipos de secciones: una sección para definir datos, una sección para código de usuario, una sección para interrupciones y una sección para interactuar con entrada/salida. Para la presente entrega no será necesario configurar la sección de interrupciones.
El simulador se distribuye como un paquete java (archivo jar) y se recomienda usar JRE 8 para ejecutarlo.

## Casos de prueba

Para la evaluación del laboratorio se utilizarán casos de prueba tanto públicos como privados. Los casos de prueba públicos se liberarán previo a la entrega del laboratorio. Los requerimientos de aprobación se publicarán más adelante.

## Información sobre la entrega - Aplicación

La aplicación a desarrollar debe ser compatible con el simulador ArquiSim v1.3.7. Se deberá entregar un programa escrito en C que modele el problema y un archivo en lenguaje ensamblador que lo implemente.

Para evaluar el funcionamiento de la aplicación debe establecerse la sección ports con la secuencia de comandos a ejecutar. En la sección de laboratorio del EVA podrán encontrar varios casos de prueba presentados como secciones ports. Para evaluar el correcto funcionamiento y el desempeño del sistema desarrollado se debe incorporar la sección ports al resto de las secciones (code, data). Ver el manual de usuario de Arquisim [1] para más información sobre el manejo de secciones.

## Información sobre la entrega - Preguntas

El día 17/10 se liberarán una serie de preguntas a contestar sobre la solución, las cuales deberán ser respondidas y serán parte de la entrega del laboratorio.

## Modalidad de trabajo

El laboratorio se realizará de forma individual.

## Forma de entrega

La entrega de la tarea debe realizarse a través del formulario habilitado en la plataforma EVA para dicho fin. La entrega del obligatorio debe ser un archivo empaquetado de nombre **obligatorio.tar.gz** u **obligatorio.zip** que debe contener los siguientes archivos:

1. un archivo C de nombre “obligatorio.c” con la implementación de alto nivel de programa
2. un archivo fuente de nombre “obligatorio.asm”, donde se considerarán los comentarios que aparezcan con el fin de facilitar la comprensión del mismo. Este archivo fuente contendrá el código assembler del programa.
3. un documento de nombre “obligatorio.pdf” con las respuestas a las preguntas planteadas.

Los trabajos deberán ser entregados indefectiblemente antes del jueves 2 de noviembre a las 23.30 horas. No se aceptará ningún trabajo pasada la citada fecha y hora. En particular, no se aceptarán trabajos enviados por correo electrónico a los docentes del curso, ni entregados en medios físicos en el instituto. El sistema de entregas soporta múltiples entregas. Pruebe de realizar una entrega con tiempo, a los efectos de verificar que su sistema le permite entregar correctamente.

## Bibliografía

[1] ArquiSim: manual de usuario. https://eva.fing.edu.uy/mod/resource/view.php?id=61253. Accedido: 12-10-2021.