Departamento de Arquitectura

Arquitectura de computadoras
Taller 8

Curso 2025

Objetivos

- Trabajar con estructuras de datos en memoria.
- Trabajar con el stack para el pasaje de parámetros.
- Trabajar con el stack para la compilación de rutinas recursivas.

Introducción

En general, cuando trabajamos con rutinas recursivas debemos almacenar temporalmente las variables locales y/o el contexto, la manera más simple de hacerlo es utilizar el stack proporcionado por la arquitectura. Cuando utilizamos el stack debemos ser consistentes al momento de guardar y quitar elementos del mismo pues un pequeño error puede tener consecuencias graves en el funcionamiento del programa.
En este taller se propone implementar una rutina recursiva, compilarla y luego realizar cálculos respecto al consumo requerido por la misma.

Ejercicio 1

Responder las siguientes preguntas.

(a) ¿Cuál es el objetivo de utilizar la secuencia de instrucciones `PUSH BP` y `MOV BP,SP`?

(b) ¿Cuál es la diferencia entre un `CALL` y un `JMP`?

Ejercicio 2

Se considera un árbol binario de búsqueda, que cumple que el subárbol izquierdo de cualquier nodo (si no está vacío) contiene valores menores que el que contiene dicho nodo, y el subárbol derecho (si no está vacío) contiene valores mayores. Se considera que los elementos contenidos en el árbol son números enteros mayores o iguales a 0, y que el elemento -1 corresponde a una hoja sin dato. Ver figura 1.

El árbol se almacenará en un Heap, es decir en un array, donde el nodo con índice i, tendrá sus hijos en los bytes con índices $2i+1$ (para el hijo izquierdo) y $2i+2$ (para el hijo derecho). El padre de un nodo, se encuentra con el índice $\left\lfloor \frac{i-1}{2} \right\rfloor$ (asumiendo que la raíz tiene índice 0). Ver figura 2.

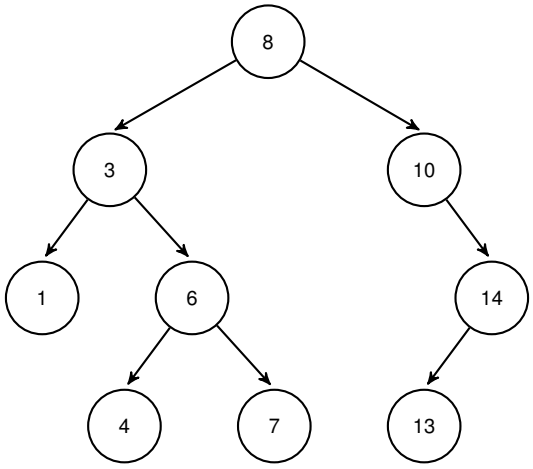

Figura 1: Árbol binario de búsqueda

A continuación se propone el código en alto nivel para realizar la búsqueda de un elemento (dado un árbol y el valor). Los parámetros de la función son: el puntero al árbol, el nodo en el cual buscar y el valor buscado.

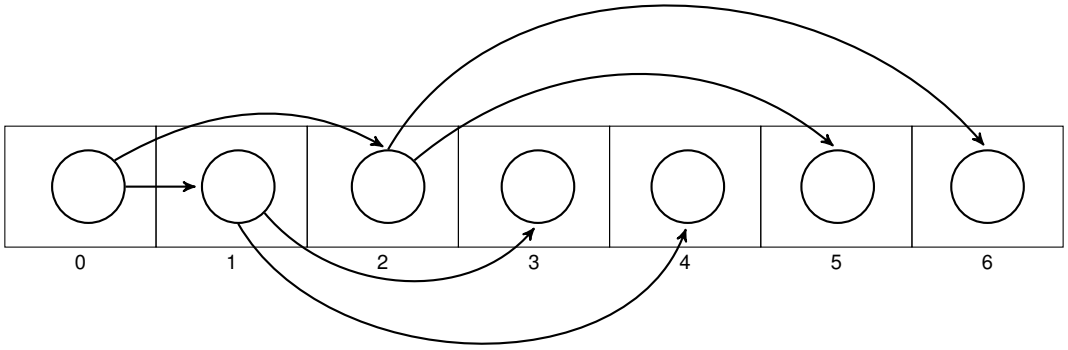

Figura 2: Árbol almacenado en un heap

```
short buscarABB(short *arbol, short indice, short valor) {
  // Paso base
  if (arbol[indice] == -1)
    return False;
  if (arbol[indice] == valor)
    return True;
  // Recursión
  else if (arbol[indice] > valor)
    return buscarABB(arbol, (2*indice)+1, valor);
  else
    return buscarABB(arbol, (2*indice)+2, valor);
}
```

(a) Compile la función presentada en assembler 8086 considerando que el array conteniendo el árbol se encuentra en el segmento ES, y los parámetros de la misma son pasados por stack. La llamada inicial y el retorno del resultado será entonces:

```
PUSH arbol
PUSH 0 ; (indice)
PUSH valor
CALL buscarABB
POP AX ; (resultado)
```

(b) Si el árbol tiene altura 10, calcule cuál es el tamaño necesario para el arreglo y cuánto espacio de stack se necesita para poder ejecutar `buscarABB` en cualquier árbol de dicha altura.

Recomendaciones generales:

- Realizar diagramas de los elementos en la memoria y de cómo se va moviendo el stack.
- Comentar el código.