Control de Subprogramas

Paradigmas de Lenguajes

3er Año LCC- Facultad CEFyN - UNSJ -

Introducción:

“Control de secuencia se refiere al mecanismo que establece la secuencia entre subprogramas”

Pautas a tener en cuenta:

- Como un subprograma **invoca** a otro subprograma.
- Como se **regresa** el control al punto de la llamada.
- Como se **comparten los datos** entre subprogramas.

https://docs.oracle.com/javase/tutorial/essential/exceptions/

Subprogramas Simples

- Llamados de enunciados simples call_return o llamada_regreso.
- Están presentes en la mayoría de los lenguajes de programación.
- Su comportamiento, en gral, es:
  - Un solo programa principal
  - Durante la ejecución puede llamar a uno/varios subprog.
  - Cada subprograma puede llamar a otro y asi sucesiva/
  - Solo uno esta en ejecución (tiene el control)
  - Debe terminar su ejecución para regresar el control
  - El subprograma que llama a otro se detiene temporalmente y puede continuar su ejecución cuando recupera el control

"Esta estructura de control se explica por la Regla de la Copia"

Subprogramas Simples

Supuestos implícitos en la Regla de la copia:

1. El subPrograma no puede ser recursivo
2. El subProg se debe ejecutar por completo en cada llamada.
3. Se requiere **call** explícito
4. Secuencia única de ejecución
5. Trasferencia inmediata del control.

“El control de secuencia en Subprogramas esta estrechamente ligado al control de datos: transmisión de parámetros, variables globales y locales”

Cada prog o subprog tiene un conjunto de asociaciones disponibles e invariantes durante su ejecución.

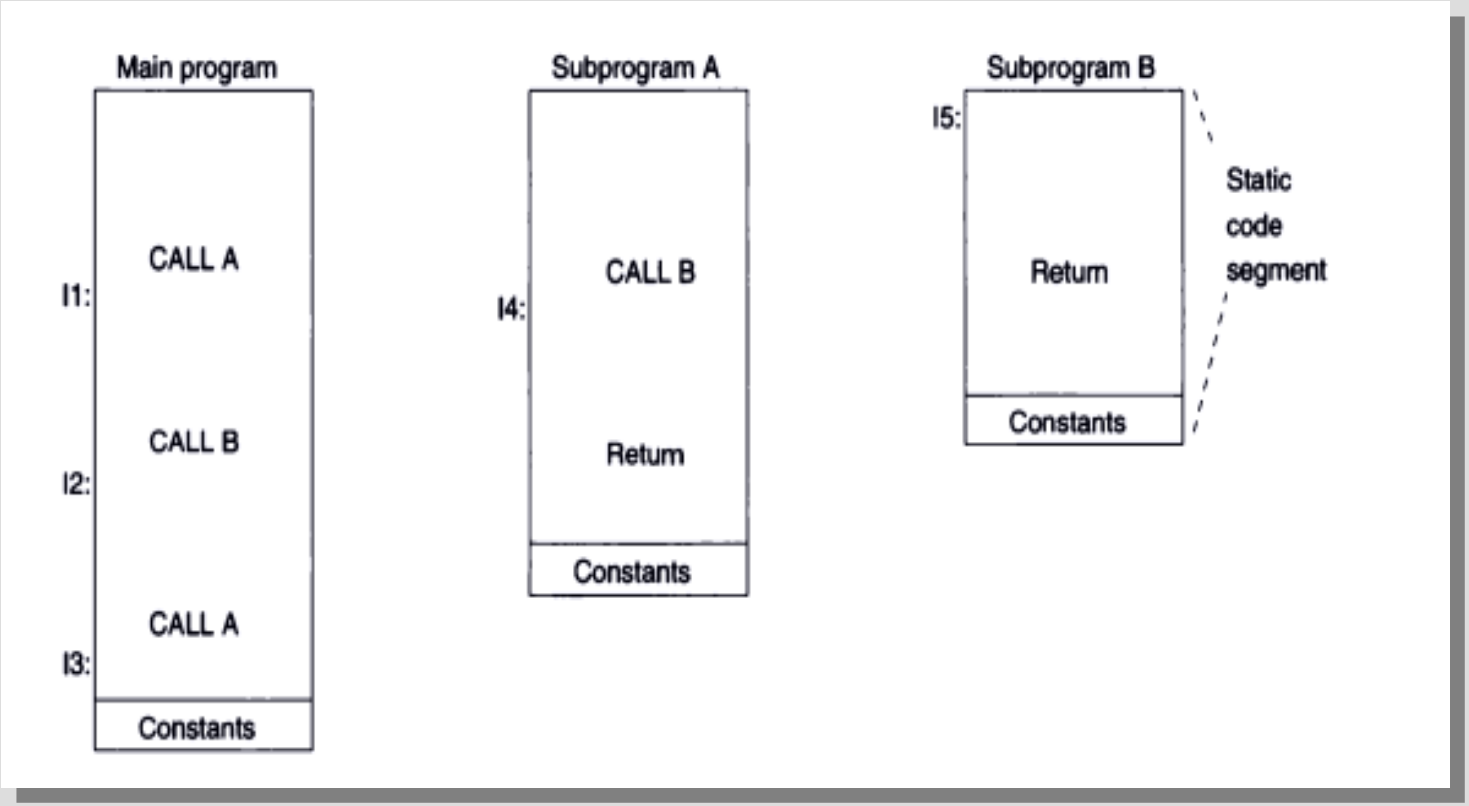
Estas asociaciones forman el **Ambiente de Referencia**: ambiente local, no local, global y predefinido. Mas la regla de alcance (estático o dinámico)

Implementación:

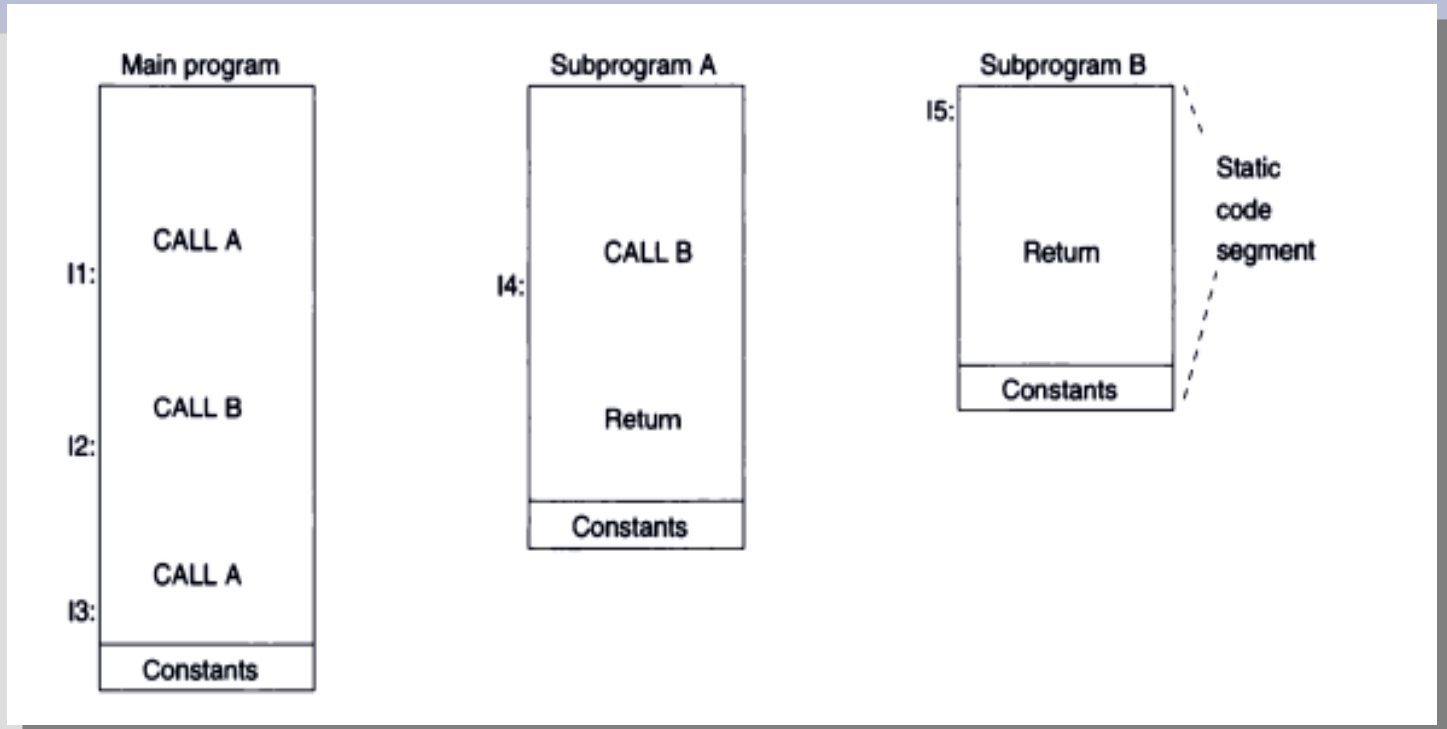
- **Definición** de subProg: es el código escrito en algún lenguaje de programación que será traducido a lenguaje de máquina (compilado o interpretado)
- **Activación** de un subProg: se crea cada vez que se invoca.
- El **segmento de código** es invariante durante la ejecución
- El **registro de activación** se crea en cada llamada y se elimina cuando retorna el control (punto de retorno, parámetros, datos locales, áreas de almacenamiento temporal, vinculación a referencias de variables no locales, resultado)

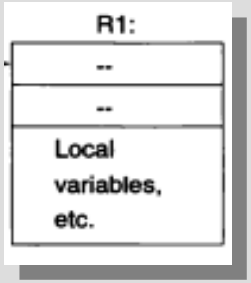
Mapa de memoria:

Segmento estático:



Estado de ejecución del principal:



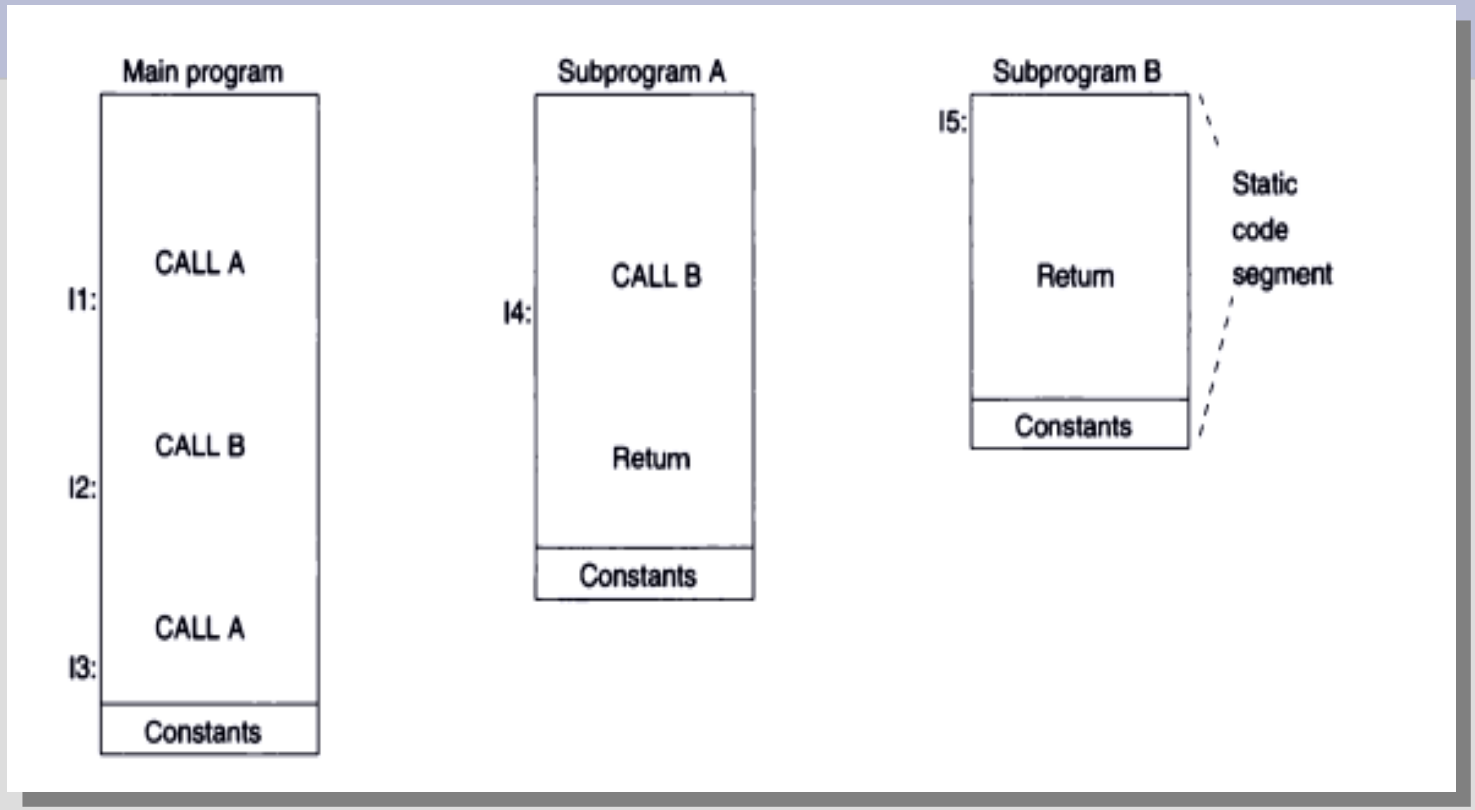


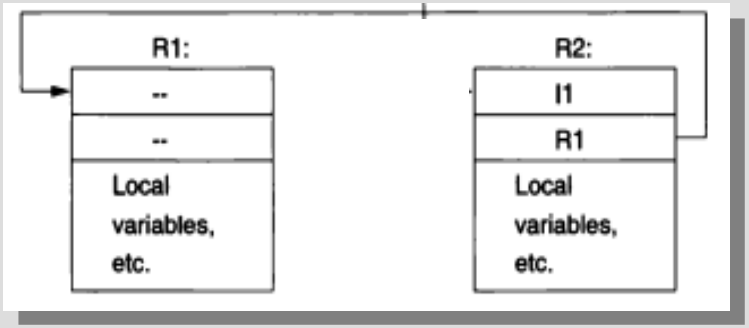
Apuntadores:

CIP: I1

CEP: R1

Estado de ejecución del Subprog B



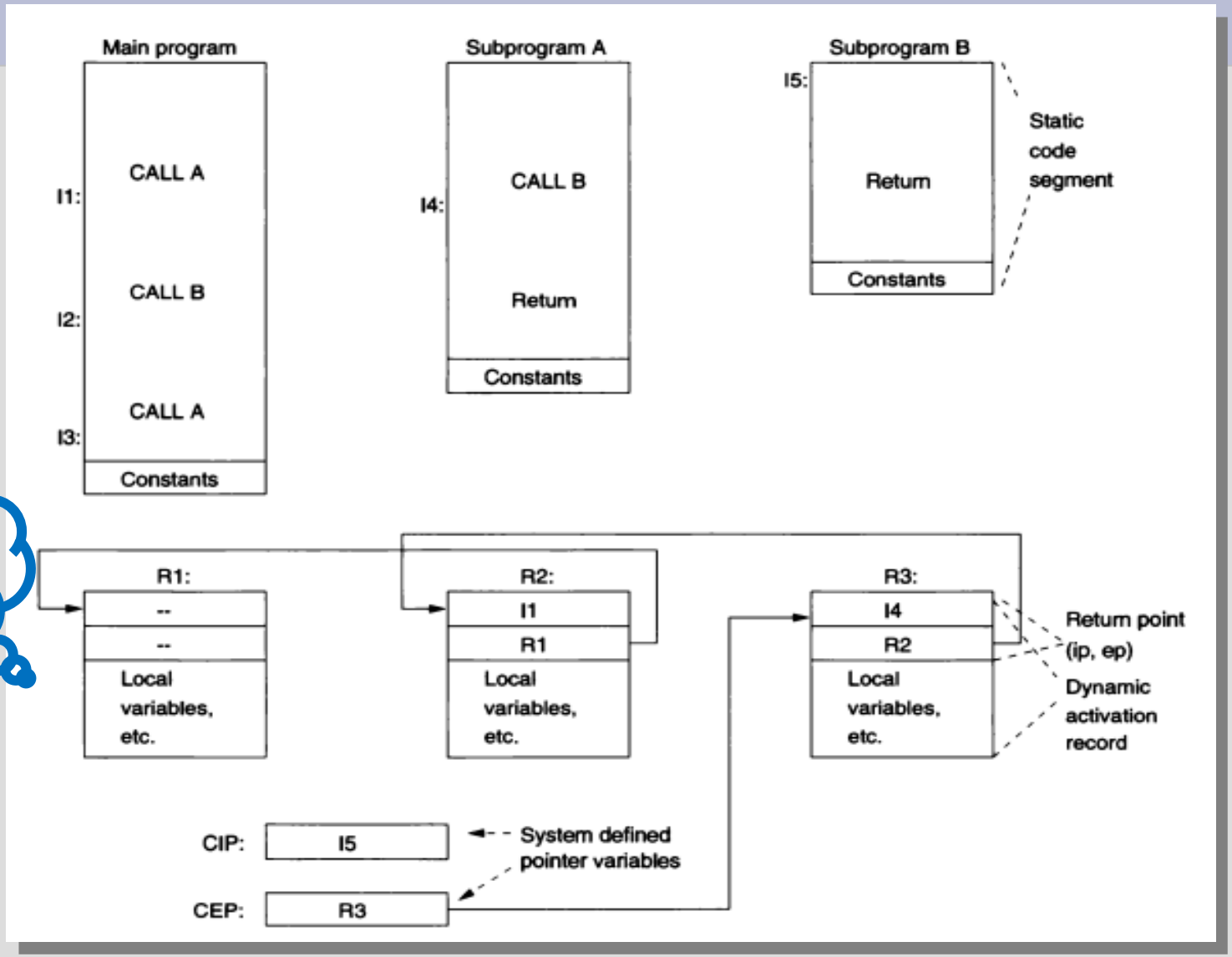


Apuntadores:

CIP: I4

CEP: R2

Estado de ejecución del Subprog B

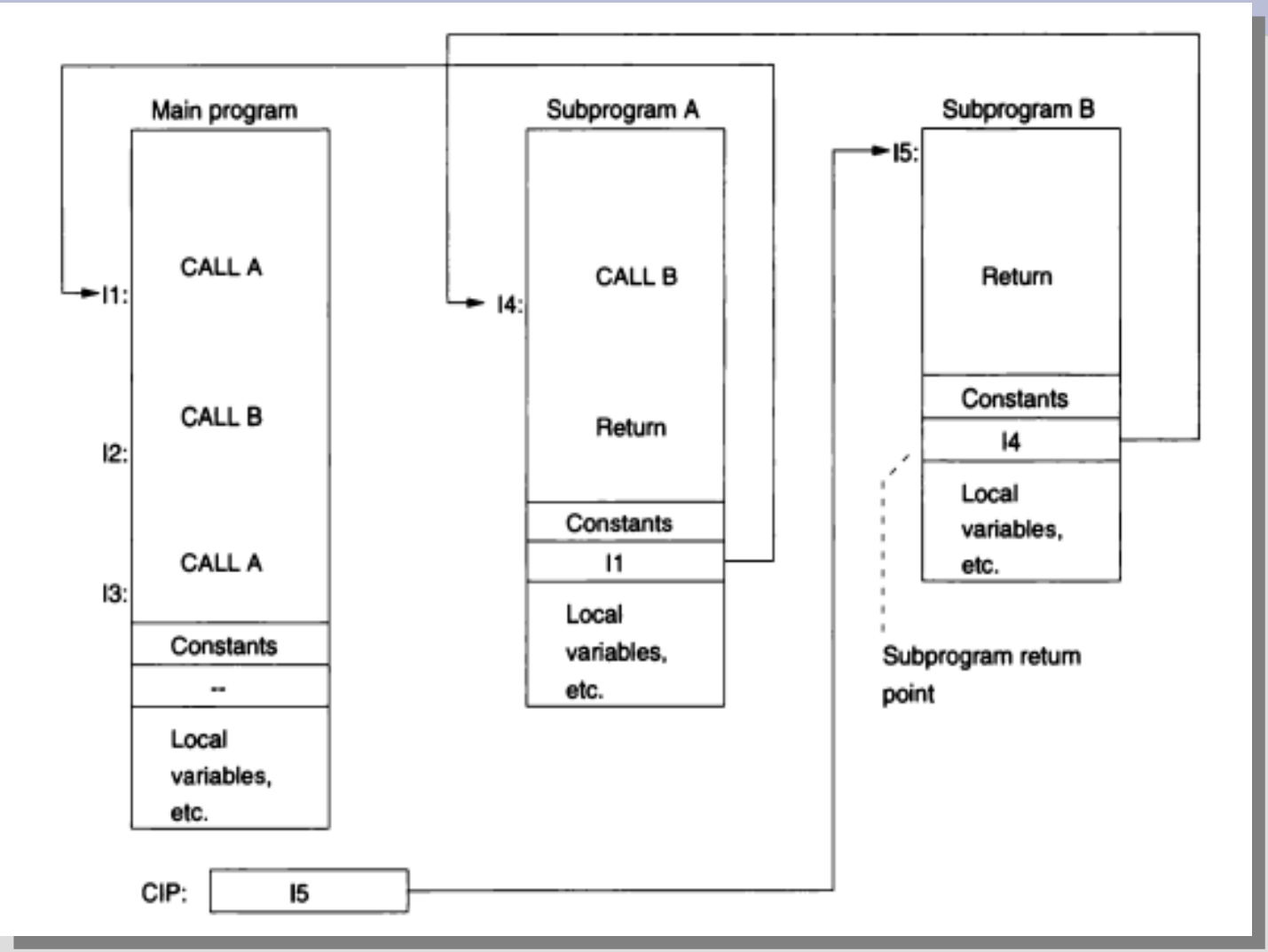


“Los Subprog Simples puede ser llamado muchas veces durante la ejecución del programa, sin embargo, tendrá solo una activación en cada llamada”

Mapa de memoria alternativo:

- Se genera un Registro de activación de cada subprog como extensión del segmento de código.
- Este Registro se almacenaría en forma estática.
- En este modelo se usa el mismo Registro en forma repetida y se inicializa en cada llamada.

Mapa de memoria alternativo



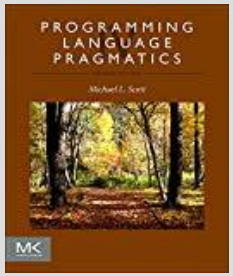
“Este modelo tiene un costo de almacenamiento y el beneficio de tiempo durante la ejecución”

Varias implementación de FORTRAN Y COBOL usan este modelo mas simple.

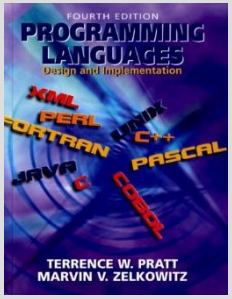
Estructuras de control complejas:

1. Recursivos
2. Corrutinas
3. Excepciones
4. Tareas
5. Planeados

Bibliografía:



Programming language pragmatics (4th Edition). 2016. Michael L. Scott.



Programming Languages: Design and Implementation (4th Edition). 2001.Terrence W.PRATT y Marvin V. ZELKOWITZ

Lenguajes de Programación. Diseño e Implementación (3ra. Edición). Terrence W.PRATT y Marvin V. ZELKOWITZ