Compiler Design (CD)
GTU # 2170701

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Unit – 6
Run Time Memory Management

Prof. Dixita B. Kagathara
Computer Engineering Department
rshan Institute of Engineering & Technology, Rajkot

dixita.kagathara@darshan.ac.in
+91 - 97277 47317 (CE Department)

Topics to be covered

- Source language issues
- Storage organization
- Storage allocation strategies
- Access to non local names
- Parameter passing
- Symbol tables
- Dynamic Storage Allocation Techniques

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Source language issues

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Source language issues

- Source language issues are:
  1. Procedure call
  2. Activation tree
  3. Control stack
  4. Scope of declaration
  5. Binding of names

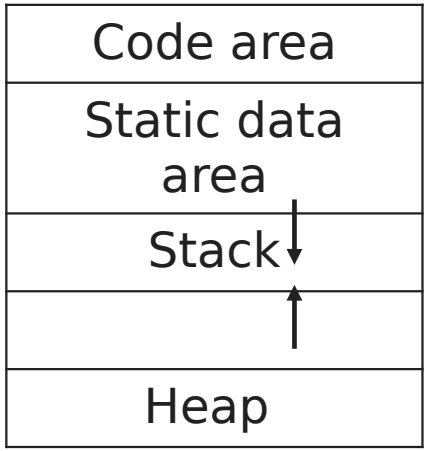
Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Storage Organization

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Subdivision of Runtime Memory

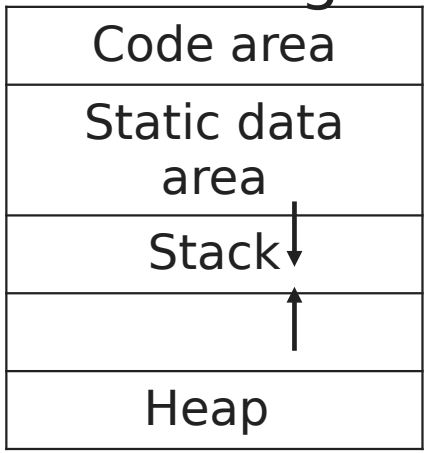
- The compiler demands for a block of memory to operating system.
- The compiler utilizes this block of memory executing the compiled program. This block of memory is called **run time storage.**
- The run time storage is subdivided to hold code and data such as, the generated target code and data objects.
- The size of generated code is fixed. Hence the target code occupies the determined area of the memory.



Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Subdivision of Runtime Memory

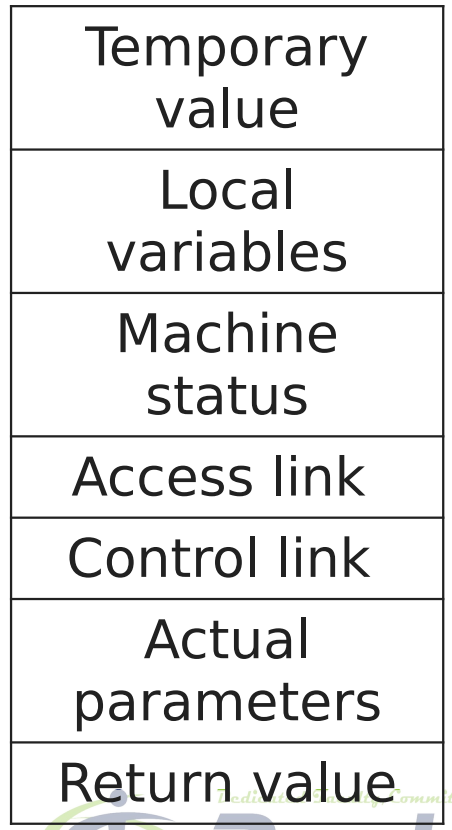
- The amount of memory required by the data objects is known at the compiled time and hence data objects also can be placed at the statically determined area of the memory.
- Stack is used to manage the active procedure.
- Managing of active procedures means when a call occurs then execution of activation is interrupted and information about status of the stack is saved on the stack.
- Heap area is the area of run time storage in which the other information is stored.



Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Activation Record

- The execution of a procedure is called its activation.
- An activation record contains all the necessary information required to call a procedure.
- **Temporary values**: stores the values that arise in the evaluation of an expression.
- **Local variables:** hold the data that is local to the execution of the procedure.
- **Machine status:** holds the information about status of machine just before the function call.
- **Access link (optional):** refers to non-local data held in other activation records.
- **Control link (optional):** points to activation record of caller.
- **Actual parameters:** This field holds the information about the actual parameters.
- **Return value:** used by the called procedure to return a value to calling procedure.



Darshan
Institute of Engineering & Technology

Compile-Time Layout of Local Data

- The amount of storage needed for a name is determined from its type. (e.g.: int, char, float…)
- Storage for an aggregate, such as an array or record, must be large enough to hold all it's components.
- The field of local data is laid out as the declarations in a procedure are examined at compile time.
- We keep a count of the memory locations that have been allocated for previous declarations.
- From the count we determine a relative address of the storage for a local with respect to some position such as the beginning of the activation record.

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Storage Allocation Strategies

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Storage allocation strategies

The different storage allocation strategies are;

- Static allocation: lays out storage for all data objects at compile time.
- Stack allocation: manages the run-time storage as a stack.
- Heap allocation: allocates and de-allocates storage as needed at run time from a data area known as heap.

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Static allocation

- In static allocation, names are bound to storage as the program is compiled, so there is no need for a run-time support package.
- Since the bindings do not change at run-time, every time a procedure is activated, its names are bounded to the same storage location.

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

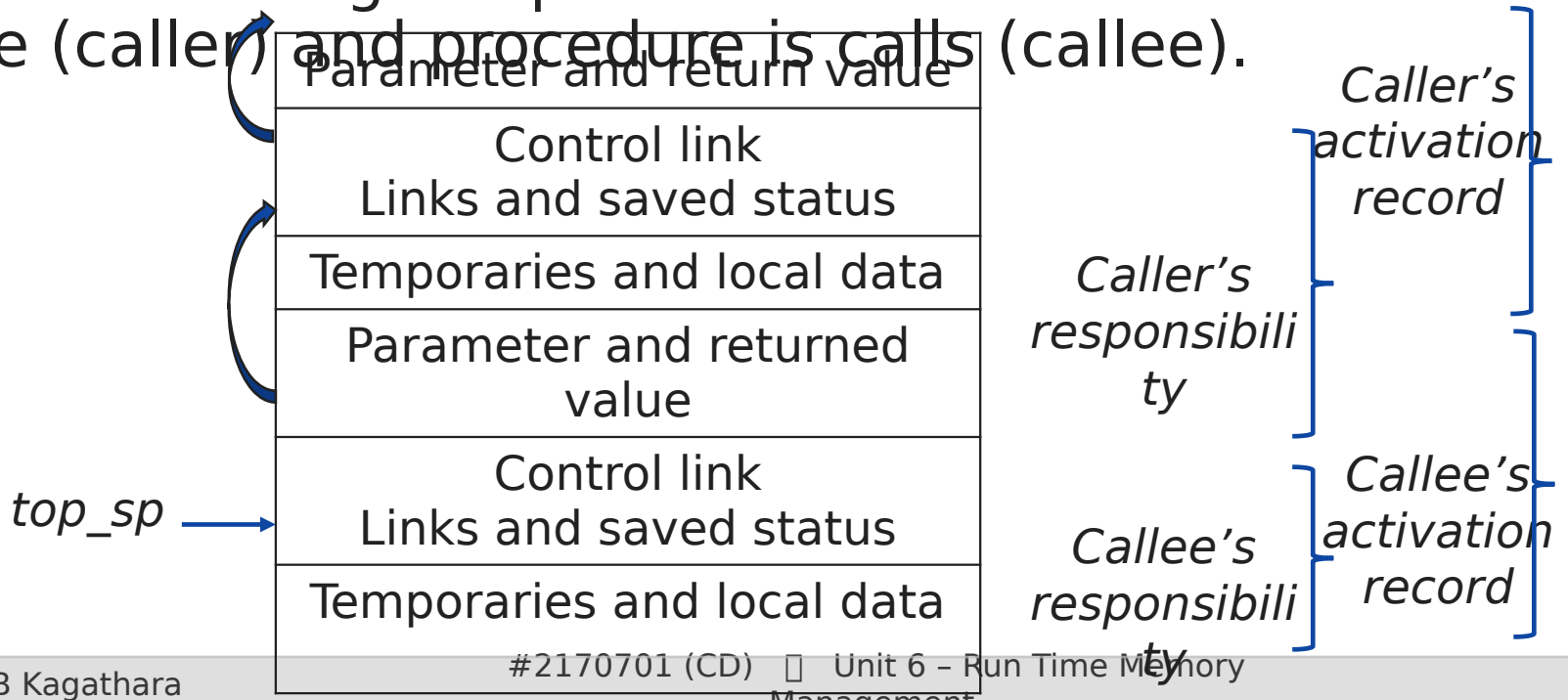
Stack allocation

- All compilers for languages that use procedures, functions or methods as units of user define actions manage at least part of their run-time memory as a stack.
- Each time a procedure is called, space for its local variables is pushed onto a stack, and when the procedure terminates, the space is popped off the stack.

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

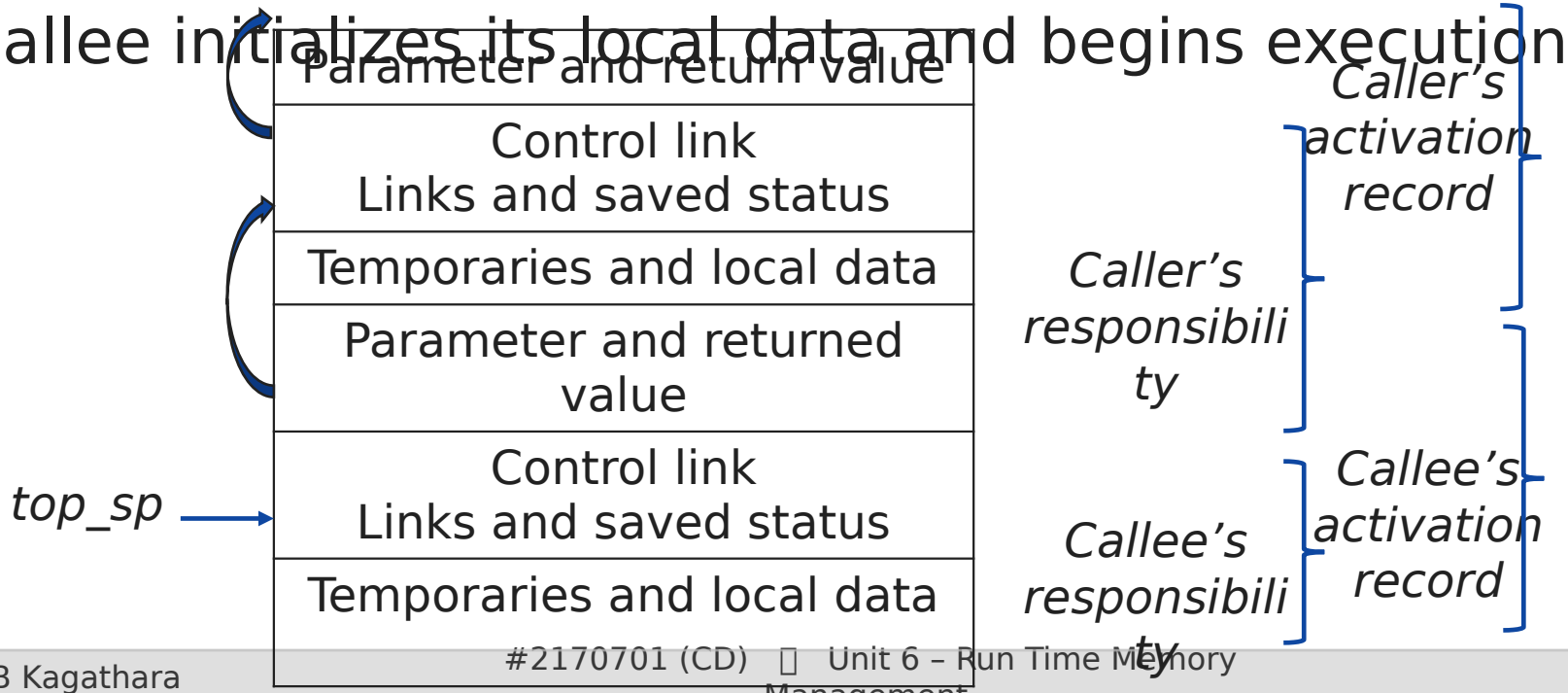
Stack allocation: Calling Sequences

- Procedures calls are implemented by generating what are known as calling sequences in the target code.
- A call sequence allocates an activation record and enters the information into its fields.
- A Return sequence restore the state of machine so the calling procedure can continue its execution.
- The code is calling sequence of often divided between the calling procedure (caller) and procedure is calls (callee).



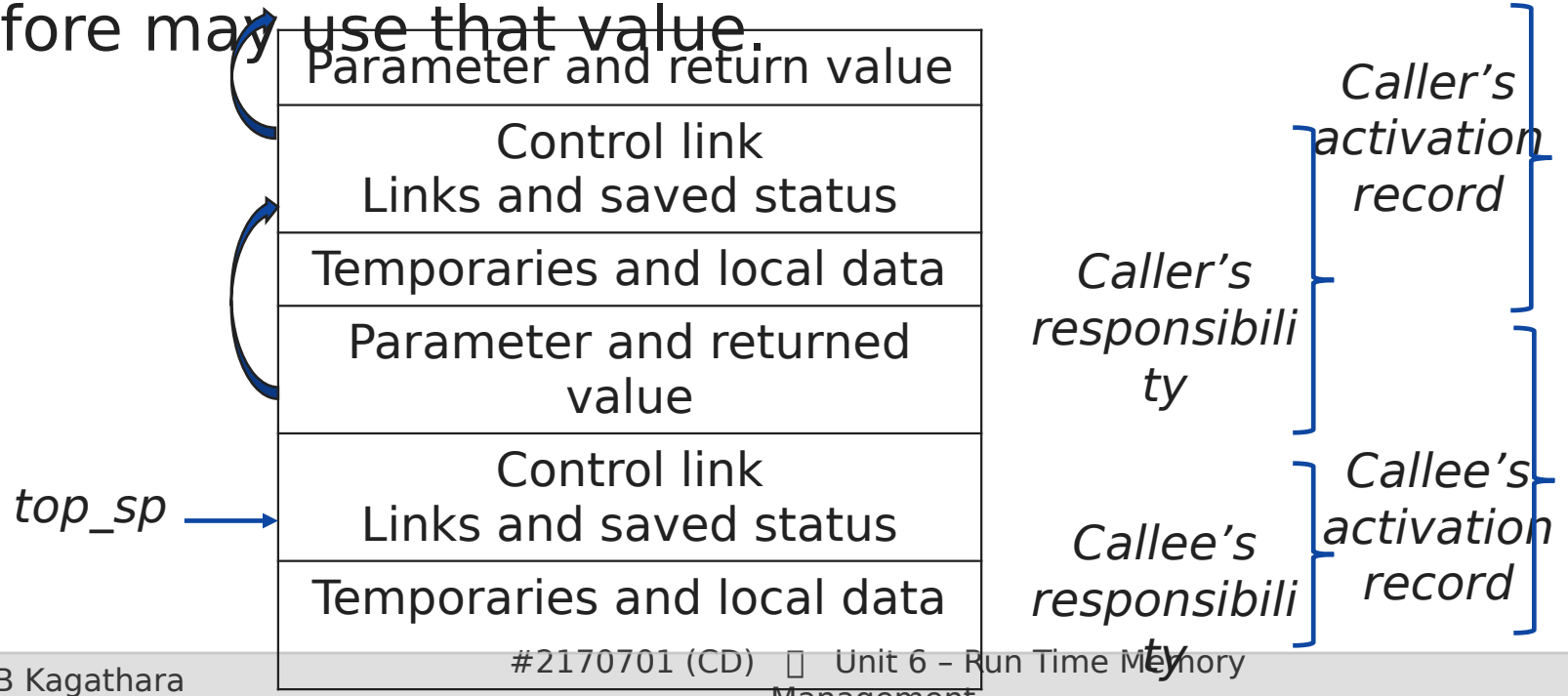
Stack allocation: Calling Sequences

- The calling sequence and its division between caller and callee are as follows:
  1. The caller evaluates the actual parameters.
  2. The caller stores a return address and the old value of *top_sp* into the callee's activation record. The caller then increments the *top_sp* to the respective positions.
  3. The callee saves the register values and other status information.
  4. The callee initializes its local data and begins execution.



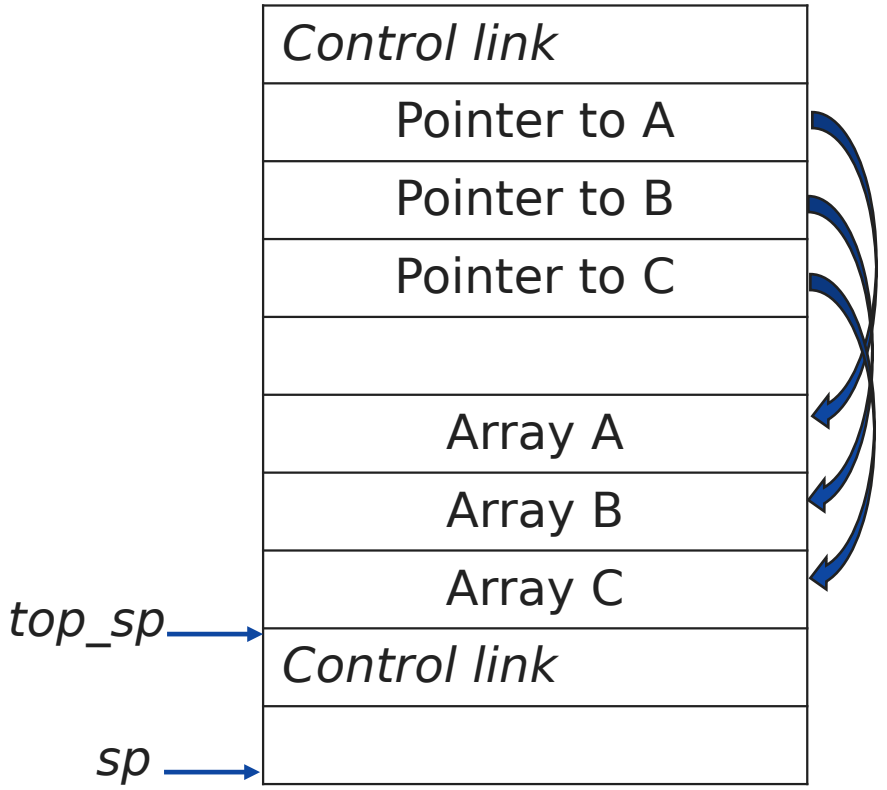
Stack allocation: Calling Sequences

- A suitable, corresponding return sequence is:
  1. The callee places the return value next to the parameters.
  2. Using the information in the machine status field, the callee restores *top_sp* and other registers, and then branches to the return address that the caller placed in the status field.
  3. Although *top_sp* has been decremented, the caller knows where the return value is, relative to the current value of *top_sp* ; the caller therefore may use that value.



Stack allocation: Variable length data on stack

- The run time memory management system must deal frequently with the allocation of objects, the sizes of which are not known at the compile time, but which are local to a procedure and thus may be allocated on the stack.
- The same scheme works for objects of any type if they are local to the procedure called have a size that depends on the parameter of the call.



Stack allocation: Dangling Reference

- Whenever storage can be allocated, the problem of dangling reference arises. The dangling reference occurs when there is a reference of storage that has been allocated.
- It is a logical error to use dangling reference, since, the value of de-allocated storage is undefined according to the semantics of most languages.

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Heap Allocation

- Variables local to a procedure are allocated and de-allocated only at runtime.
- Heap allocation is used to dynamically allocate memory to the variables and claim it back when the variables are no more required.
- Except statically allocated memory area, both stack and heap memory can grow and shrink dynamically and unexpectedly.
- Therefore, they cannot be provided with a fixed amount of memory in the system.

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Access to Non local names

- A procedure may sometimes refer to variables which are not local to it.
- For the non local names the rules can be defined as: static and dynamic

Static scope rule

- The static scope rule is also called as lexical scope.
- Scope is determined by examining the program text.
- PASCAL, C and ADA are the languages use the static scope rule.
- These languages are also called as block structured language.

Dynamic scope rule

- For non block structured languages this dynamic scope allocation rules are used.
- The dynamic scope rule determines the scope of declaration of the names at run time by considering the current activation.
- LISP and SNOBOL are the languages which use the dynamic scope rule.

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Parameter Passing Methods

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Parameter Passing Methods

- There are two types of parameters, Formal parameters & Actual parameters.
- And based on these parameters there are various parameter passing methods, the common methods are:
  1. Call by value
  2. Call by reference
  3. Copy restore
  4. Call by name

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Call by Value

- This is the simplest method of parameter passing.
- The **call by value** method of passing arguments to a function copies the actual value of an argument into the formal parameter of the function.
- The operations on formal parameters do not change the values of a parameter.

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Call by Reference

- This method is also called as call by address or call by location.
- The **call by reference** method of passing arguments to a function copies the address of an argument into the formal parameter.
- Inside the function, the address is used to access the actual argument used in the call.
- It means the changes made to the parameter affect the passed argument.

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Copy Restore

- This method is a hybrid between call by value and call by reference.
- This method is also known as copy-in-copy-out or values result.
- The calling procedure calculates the value of actual parameter and it then copied to activation record for the called procedure.
- During execution of called procedure, the actual parameters value is not affected.
- If the actual parameter has L-value then at return the value of formal parameter is copied to actual parameter.

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Call by Name

- This is less popular method of parameter passing.
- Procedure is treated like macro.
- The procedure body is substituted for call in caller with actual parameters substituted for formals.
- The local names of called procedure and names of calling procedure are distinct.
- The actual parameters can be surrounded by parenthesis to preserve their integrity.

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Symbol Table

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Symbol Table

- Symbol table is a data structure used by compiler to keep track of semantics of a variable.
- Symbol table is built in lexical and syntax analysis phases.
- The items to be stored into symbol table are:
  1. Variable names
  2. Constants
  3. Procedure names
  4. Function names
  5. Literal constants and strings
  6. Compiler generated temporaries
  7. Labels in source language

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Data structures for a symbol table

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

List Data structure

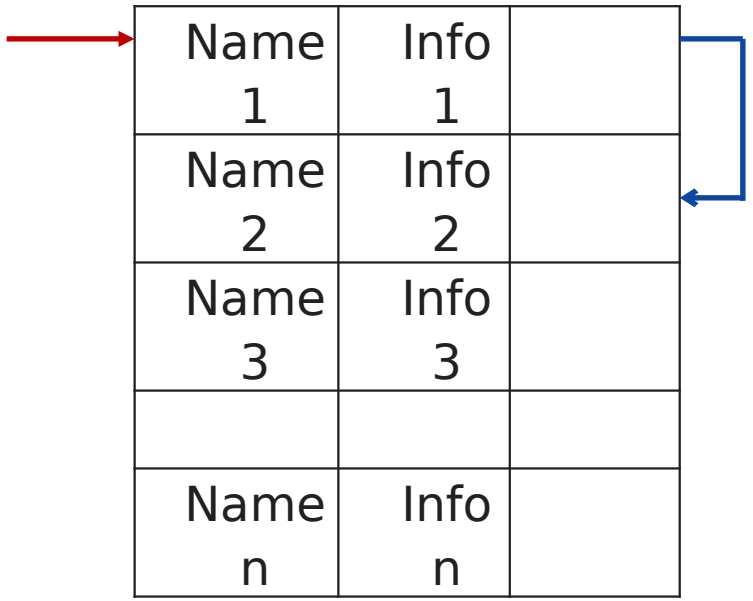
- The name can be stored with the help of starting index and length of each name.
- Linear list is a simplest kind of mechanism to implement the symbol table.
- In this method an array is used to store names and associated information.
- New names can be added in the order as they arrive.
- The list data structure using array is given below:

Name 1	Info 1
Name 2	Info 2
Name 3	Info 3
Name n	Info n

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Self organizing list

- This symbol table implementation is using linked list. A link field is added to each record.
- We search the records in the order pointed by the link of link field.
- The pointer “First” is maintained to point to first record of the symbol table.



Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Binary tree

- When the organization symbol table is by means of binary tree, the node structure will as follows:
- The left child field stores the address of previous symbol.
- Right child field stores the address of next symbol.
- The symbol field is used to store the name of the symbols.
- Information field is used to give information about the symbol.

Left child	Symbols	Information	Right child

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Hash table

- In hashing scheme two tables are maintained-a hash table and symbol table.
- The hash table consists of k entries from 0,1 to k-1. These entries are basically pointers to symbol table pointing to the names of symbol table.
- To determine whether the 'Name' is in symbol table, we use a hash function 'h' such that h(name) will result any integer between 0 to k-1. We can search any name by position=h(name).
- Using this position we can obtain the exact locations of name in symbol table.
- Advantage of hashing is quick search is possible and the disadvantage is that hashing is complicated to implement.

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Dynamic Storage Allocation Techniques

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Dynamic Storage Allocation Techniques

- There are two techniques used in dynamic memory allocation.

1. Explicit allocation
2. Implicit allocation

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

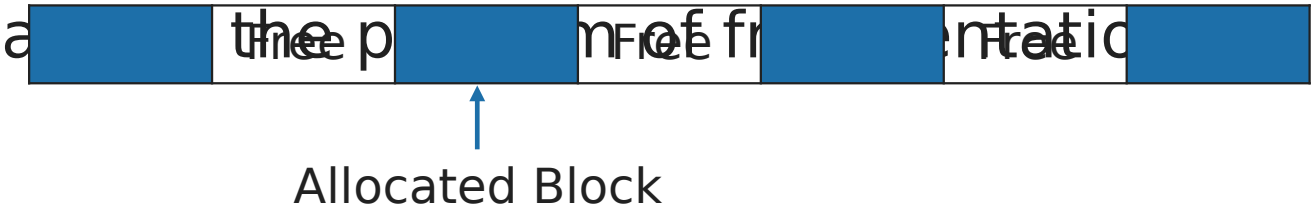
Explicit Allocation: for Fixed Size Blocks

- In explicit allocation the size of the block for which memory is allocated is fixed.
- In this technique a free list is used. Free list is a set of free blocks.
- The blocks are linked to each other in a list structure. The memory allocation can be done by pointing previous node to the newly allocated block.
- Memory de-allocation can be done by de-referencing the previous link.
- This memory allocation and de-allocation is done using heap memory.
- The advantage of this technique is that there is no space overhead.

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

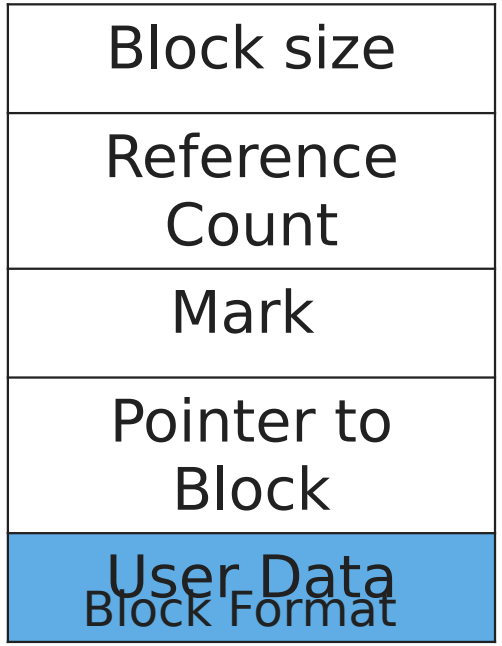
Explicit Allocation: for Variable Sized Blocks

- Due to frequent memory allocation and de-allocation the heap memory becomes fragmented.
- That means heap may consist of some blocks that are free and some that are allocated.
- Thus we get variable sized blocks that are available free.
- For allocating variable sized blocks some strategies such as first fit, worst fit and best fit are used.
- Sometimes all the free blocks are collected together to form a large free block.
- This ultimately a



Implicit Allocation

- The implicit allocation is performed using user program and runtime packages.
- The run time package is required to know when the **storage block** is not in use.



Block Format

Implicit Allocation: Reference count

- Reference count is a special counter used during implicit memory allocation.
- If any block is referred by some another block then its reference count incremented by one.
- That also means if the reference count of particular block drops down to 0 then, that means that block is not referenced one and hence it can be de-allocated.
- Reference counts are best used when pointers between blocks never appear in cycle.

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Implicit Allocation: Marking techniques

- This is an alternative approach to determine whether the block is in use or not.
- In this method, the user program is suspended temporarily and **frozen pointers** are used to mark the blocks that are in use.
- Sometime bitmaps are used to the blocks that are in use.
- These pointers are then placed in the heap memory.
- Again we go through heap memory and mark those blocks which are unused.
- Using marking technique it is possible to keep track of blocks that are in use.

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Thank You

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology