STORAGE ALLOCATION STRATEGIES

1. Static Allocation: It is for all the data objects at compile time.
2. Stack Allocation: In this a stack is used to manage the run time storage. For example recursive calls make use of this area.
3. Heap Allocation: In this a heap is used to manage the dynamic memory allocation.

Static Allocation

- The size of data objects is known at compile time.
- The names of these objects are bound to storage at compile time only and such an allocation of data objects is done by static allocation.
- The binding of name with the amount of storage allocated do not change at run time. Hence the name of this allocation is static allocation.
- In this the compiler can determine the amount of storage required by each data object and therefore it becomes easy for a compiler to find the address of these data in the activation record.
- At compiler time compiler can fill the address at which the target code can find the data it operates on.
- FORTRAN uses the static allocation.

Limitation of static allocation

- The static allocation can be done only if the size of data object is known at compile time.\
- The data structures cannot be created dynamically. In the sense that, the static allocation cannot manage the allocation of memory at run time.
- Recursive procedures are not supported by this type of allocation.

Stack Allocation

- In this the storage is organized as stack .This stack is also called control stack.
- As activation begins the activation records are pushed onto the stack and on completion of this activation the corresponding activation records can be popped.
- The locals are stored in the each activation record. Hence locals are bound to corresponding activation record on each fresh activation.
- The data structures can be created dynamically for stack allocation.

Limitation of stack allocation

The memory addressing can be done using pointers an index registers. Hence this type of allocation is slower than static allocation.

Heap Allocation

- If the values of non local variables must be retained even after the activation record then such a retaining is not possible by stack allocation. This limitation of stack allocation is because of its Last in First Out nature. For retaining of such local variables heap allocation strategy is used.
- The heap allocation allocates the continuous block of memory when required for storage of activation records or other data object, this allocated memory can be deallocated when activation ends. This deallocated space can be further reused by heap manager.
- The efficient heap management can be done by
- Creating a linked list for the free blocks and when any memory is deallocated that block of memory is appended in the linked list.
- Allocate the most suitable block o memory from the linked list i.e. use best fit

technique for allocation of block.

Related Posts:

1. Introduction to Compiler
2. Analysis and synthesis model of compilation
3. Bootstrapping and Porting
4. Lexical Analyzer: Input Buffering
5. Storage Allocation Strategies
6. Type Checking
7. Specification & Recognition of Tokens
8. Front end and back end of the compiler
9. LEX
10. Analysis synthesis model of compilation
11. Data structure in CD
12. Register allocation and assignment
13. Loops in flow graphs
14. Dead code elimination
15. Syntax analysis CFGs
16. L-attribute definition
17. Operator precedence parsing
18. Analysis of syntax directed definition
19. Recursive descent parser
20. Function and operator overloading
21. Equivalence of expression in type checking
22. Storage organization
23. Parameter passing

24. Run time environment
25. Type checking
26. Code generation issue in design of code generator
27. Boolean expression
28. Declaration and assignment in intermediate code generation
29. Code optimization
30. Sources of optimization of basic blocks
31. Loop optimization
32. Global data flow analysis
33. Data flow analysis of structure flow graph (SFG)