

Indian Institute of Technology
Kanpur
NP – TEL
National Programme
On
Technology Enhanced Learning
Course Title
Compiler Design
Lecture – 27

by...

Prof. S.K. Aggarwal.

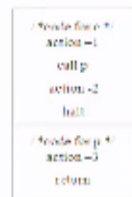
Dept. of computer Science and Engineering.

So what we are doing assume this point of time that a high 4 types of instructions one is call be return.

(Refer Slide Time: 00:30)

Run Time Storage Management

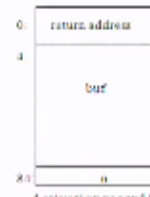
- Run time allocation and de-allocation of activations occurs as part of procedure call and return sequences
- Assume four kind of statements:
call, return, halt and action



Three address code



Activation record for c (64 bytes)

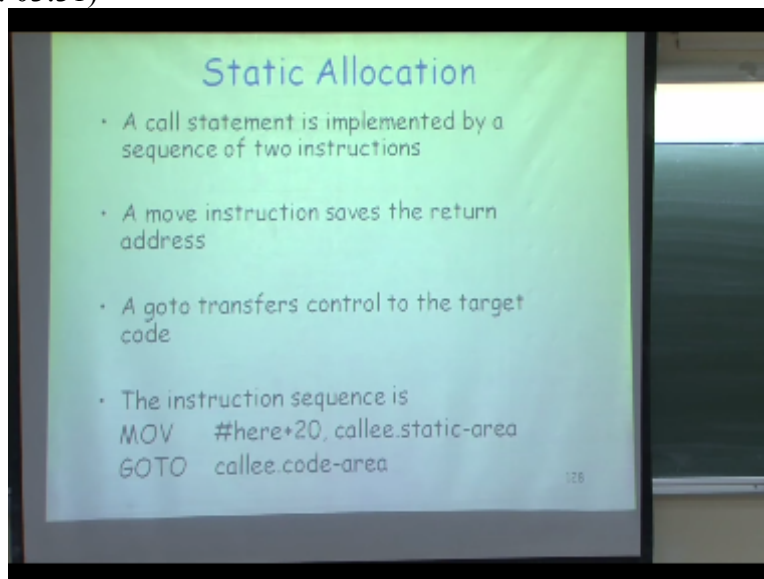


Activation record for p (88 bytes)

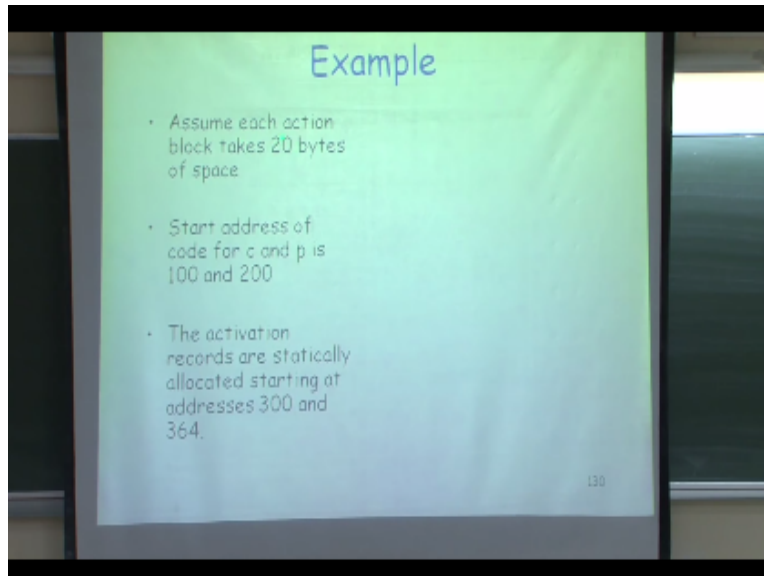
And reductions are statement which is in order procedure of function of final statements. To run time is orient though control so on okay and we do not have worry about this but very important part is very well oriented. So we look at statically location now what will have do is allocation location is be cores what to the physical form and is will be part return storage. So let us assume two functions one is see another is function C and P and C is going to info P by having a statement and our connection.

To our some pieces of ports on any innovation so all statement is will be manipulating my return addresses I have to in so for example what I need to do is like I said on E I need this so when I call me that goes here when the ball comes back it is not at the same point part of the instruction is next t fall so this is what we move instructions you have a return address and then a boat boys will transfer control to the target what that means when I say all P first thing that will happen is

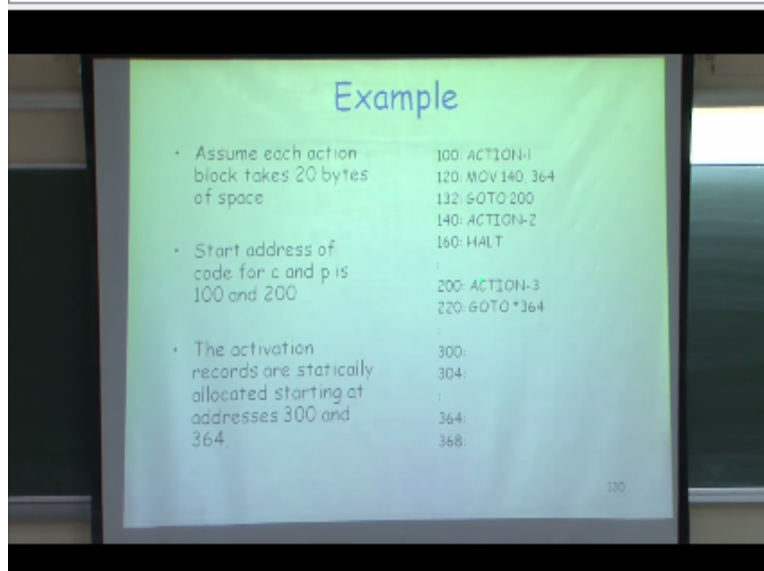
that this return this and this is moving in make certain assumptions here and then and that means if I start counting how many words this instruction will take this instruction. Will take one but for this awkward one word for this address and one word for three seconds and similarly DC make 4 pi X and this would be low but support will if you come I say or D by step so whatever is my current atmosphere who is going then the jump return and this has to be instruction which is next to this address so this is what we do and then we say that all the static data and the static area of the constants because I am talking about static allocation. (Refer Slide Time: 03:31)



So their effort to address and therefore while you get this is my rifle you put we are done it so modest move instruction is going is we saying that smoke this addressed to the policy attic area and policy attic is obviously the address of the activation record of so let me straight away go to the court. (Refer Slide Time: 05:10)



And maybe put certain number so let us assume that each action will take 40 bytes and start address off the board address of CMPs an activation reports statically will be located in the locations is So we do something like this they add this code.
 (Refer Slide Time: 05:39)



Corresponding to P hit an area of C and a great idea of great ideas so what do I do now so if you look at this instruction which says move 142 364 so what this is saying is that whatever is the activation record to support this address which is really the return address so that when this activation returns control and then at this location I say 200 and what is the 200 is the core area. So we can be more and when return happens what will return us a return to saying both of the address which is called in physics before it was PCP for physics before is the first address of the activation of P so now when I say go to 364 I already scored 140 here that means I will jump

here that 140 will assume. So let us go back to basically what we are saying is that the call static area and call code area constant activation of the first address of procedure code area respectively.

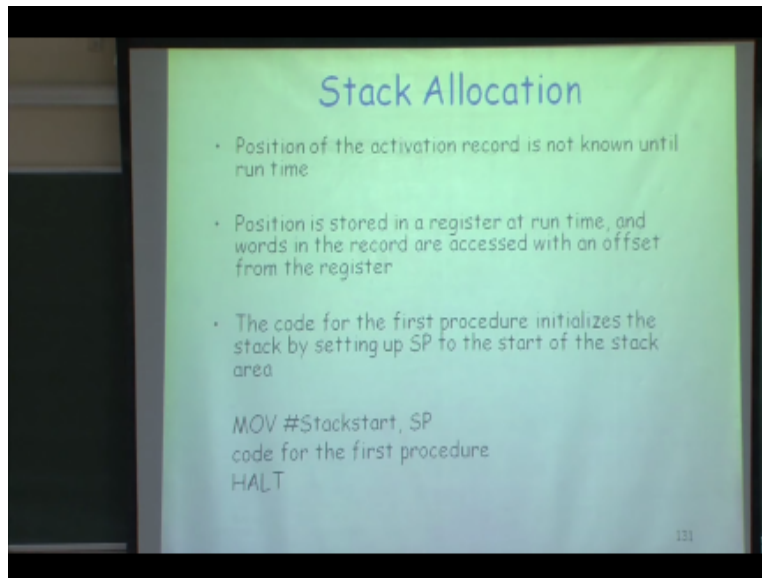
Which is this is of course I have instructions here so we discussed also instructions for open sales machine status so for example what I do here is that actually when I say move this visible news editor at this walk this I also have two instructions for setting up activation record which we discussed know some instructions for setting up all the accessories and we also have to have instructions for having the education of those unbelievers so static media is steady in location is fairly straightforward I know all the addresses of divisions and once I fix all these at this company.

And also see that what also means is that when I am talking about all these addresses starting from 100 what this means is hundred is the base address which is reloadable address and actually code gets loaded if it gets executed all the addresses will be changed with respect to the space so when I say 140 this is with respect to this and if this base thing is to some other location then obviously this will also change back normally this stuff thank you any questions on this part okay it is not so what I need to also do is.

When I say that I have this small debt so I have implemented all through two instructions one is that storing the return address in time job to the code in the but actually call it the sequence of several instructions so one is to create access links on each to save the state of the business hey that means I also have to have a sequence of instructions here could you say move register r1 to certain local area move register 2 to certain locality and so on now I have to decide which registers to say so suppose I have a to this person I want to save that in the activation here.

So I also have to have these instructions before I jump to 200 it started putting it I have instruction here which will say that save all these into the temporary locations which are part of this activation and similarly when I say go to this before this one who I also have to have instructions which you say that move all the data which I save get back into the resistance so I have to save the motivators and we will have to the call can we go back to what we did in the what this we discussed move over but if not us so this is what I do in static allocation now that is the new status that we have some code area.

(Refer Slide Time: 10:56)



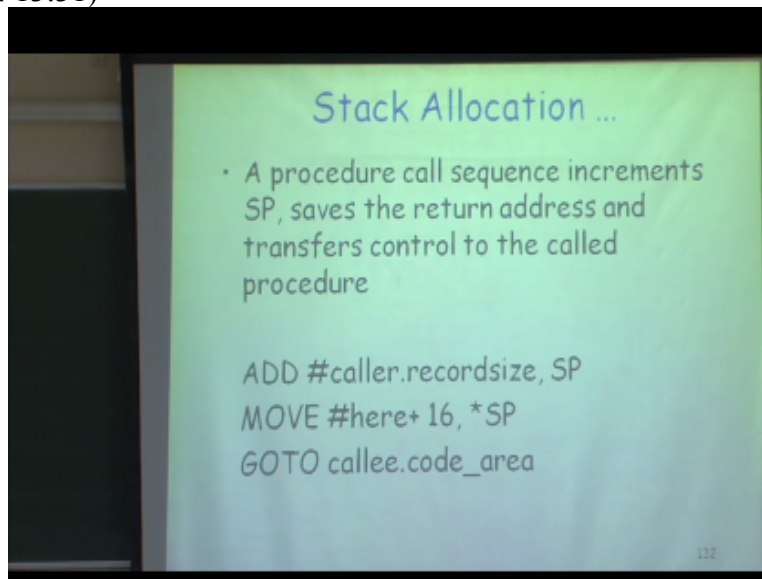
So this is all the code and all the functions and procedures okay and I also have now what I can assume is that even for the main programs my stack suppose may start from here and this is the static data or the global data. And this is how my stack will start to so first thing I need to remember there is that what is my stack pointer now normally what we do is there mornings what is making but let us work with one pointer from the time eat at one point that is the stack pointer and the stack pointer this begin which may be here and then I say that activation is on top of stack pointer since.

I know the size of activation record or in this case mobile data I can always find out with respect to the stack pointer there is the offset so I also be to therefore remember that what is the location of which I will start loading my stack so what we do here is that position of activation record is not until the runtime until your program is executing I will not know where in memory this activation record is good because depending upon the call sequence this activation to the anywhere because also remember that so position is going to be stored in the register.

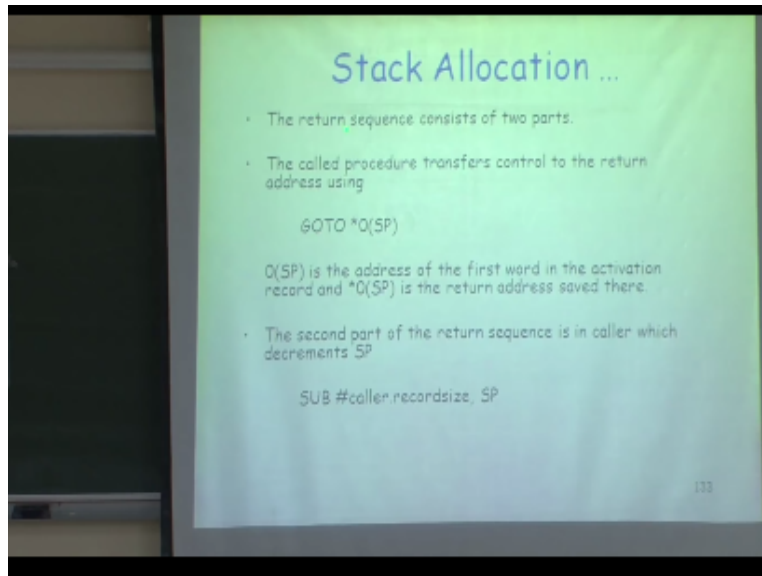
So I have suffocation we do not say that what is my stack pointer and that value must be so and at runtime then the stack pointer sorting happening that I say initially this may be my stack pointer than I wrote this up here okay now with respect to the stack pointer I can find out all the variables here and now then a new excavation comes then I have been given my stack pointer by saying that stack pointer plus side of this exhibition because the new value of the stack pointer and a new activation comes here right so this is how my stack I only have to initialize it once again.

So this is that before I even start executing my first function I say that some spark addresses from where this back we started start coding and this is really the fault destruction so what we know is

that every time pushing a sign of the activation and when I return the reverse action has to take place where I say that the stack pointer gets decremented by size.
(Refer Slide Time: 13:31)

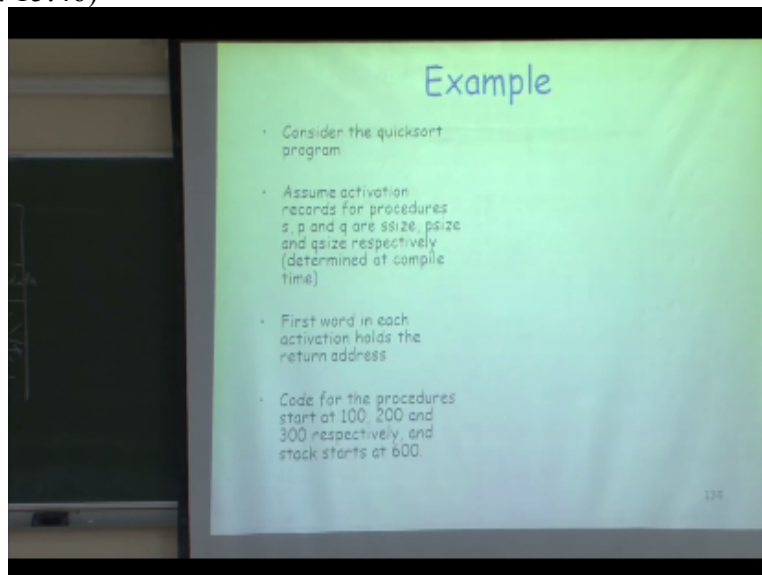


So now what happens here is I say that all sequences now implements that conscience a return address and transfer them controls so this is the new thing earlier about the saving the instance will control but now I also have to implement my step 1 so all in my inscription slow I'd say that add to stack pointer the coloring or sense so assumption made is that the stack pointer is at the bottom of this and then I say move here + 16 into the location which is want you to buy stock market that means into the location once I have implemented this is. Where the first location of the activation is and that is where I store my return address so same strategy as I using earlier that return addresses always in the bus location in this okay in this case since I do not know that but I mean say the assumption that each of coordinate it is going to be stable for bites but in this is since this is a distal argument I need only four bytes for this off port four bytes from this address four bytes for this awkward position so this is normal all set sequence.
(Refer Slide Time: 15:07)



What if I return sequence the return sequence is just saying that go to whatever is pointed to by stack pointer with an offset of zero. So this is a standard thing yeah what we say here is that this basically this is saying that if in just in case my return address is somewhere else then I can use to see that value here but basically this is saying that whatever is the location point you to buy stack pointer that is where my control desk over and obviously then you have to subtract from stack point or whatever is there all of size of that activation. So how do you my code now look so let us go?

(Refer Slide Time: 15:46)



Program I have this you saw so far was the main program so we can make certain assumptions here that the size of activation records are if I say sort partition in quick sort they are given by number S size P size and Q size produces and this become with some I because on each

activation I know how many activity I may not know the location because of a location expected location but at least I know the sizes so these sizes are my pen and first part of each of these and we also assume that the code area in the four area the procedures they started 100, 200, and 300 and stack started 600 okay.

Already given it and stack started but all these information i know is going to be available to me what I did not know is that in which order I am going to pushed activations and therefore that for this to be generated as part of my all singles so for me my code looks something like this.
(Refer Slide Time: 17:06)

Example

<ul style="list-style-type: none"> • Consider the quicksort program • Assume activation records for procedures s, p and q are ssize, psize and qsize respectively (determined at compile time) • First word in each activation holds the return address • Code for the procedures start at 100, 200 and 300 respectively, and stack starts at 600. 	<pre> action-1 /* code for s */ call q action-2 halt action-3 /* code for p */ return action-4 /* code for q */ call p action-5 call q action-6 call q return </pre>
--	---

134

So let us say that poor looks something like this that in the main program for salt I want to fix or so there may be some action here if it is inclusive election then I have a call to queue and then I have another non-postal election is an S and in P so which is partition I just take some actions and return and in case of quick sort I have some action then I have a recursive call to speed and some direction so in this I have a call to P and I have to recursive calls to so when I call quick sort I first partition.

And then I so this is how my create this for code looks but when I go to the Machine kind of work here let us understand so this is the total area which has been given to by so i push that value in sequent so 600 okay I would written in 600 is,
(Refer Slide Time: 18:11)


```

100: MOVE #600, SP
108: action-1
128: ADD #ssize, SP
136: MOVE 152, *SP
144: GOTO 300
152: SUB #ssize, SP
160: action-2
180: HALT
...

200: action-3
220: GOTO *(SP)
...

300: action-4
320: ADD #qsize, SP
328: MOVE 344, *SP
336: GOTO 200
344: SUB #qsize, SP
352: action-5
372: ADD #qsize, SP
380: MOVE 396, *SP
388: GOTO 300
396: SUB #qsize, SP
404: action-6
424: ADD #qsize, SP
432: MOVE 448, *SP
440: GOTO 300
448: SUB #qsize, SP
456: GOTO *(SP)

```

To stack month and then have this action let me say that and mouth since this is going to make thought this is I am saying that act size of S code stack pointer so my stack pointer gets implemented then I say move here + 16 into the location which is pointing device type pointer so I move onto 1:36 + 16 which is here that is where the control will come and then I say jump to 300 so where will this return address next or so return address will gets code area.

Into this is what my stack pointer started dude thank you to the first location here and then this is where I support value of this is my first phone into this pot it so what happens if in this part I am saying that now I am going to make call to first I am going to make a call to P so what I do is I say that it is size of Q and push that to the stack pointer so I'm not going to say that this is the activation corresponding and then the next exhibition has to be pushed so again so this is I increment by the size of the caller and then I say move 344 into the location which is pointed to by stack pointer.

So what is 344 is 320 + 60 and then I say now jump to location 200 and is the area and what does this to this office is when whenever the return happens a return happens by looking at Whatever is my current stack pointer and this which is called in the location right in officer it means the first location of pipe activation jump to that so whatever yes I support there and every time you said this would be different depending upon the code procedure but does not matter fourth addresses I only jump in making any direct I am saying that whatever is in this location and every time you will see that in this case I am sorry 344 here in this case times 49.

So when I now make sure it does it all here I am going to say that increment now this buy two size. Once again move 396 which is three 80+ 16 into the stack pointer and then jump to 300 so 300 is now and once the control comes back will come back this location which will say that

now subtract the spot and then I come forth then again I am going to add two sides quest and then I say move 48 to this for $32 + 16$ it is for dislocation jump to 300 which is a recursive call and comes back I come here and finally.

I jump return from this procedure and what is the language which are stored here that friend you must have been initialized at this location so there is a for IDs to and they you can see that this code this is the code really now it is magnetic. So irrespective of what like all sequences and what is the value which is to be returned by partition this code is going to remain the same it only thing that will happen is that we look at the section 5 reduction 6 they are going to determine whether I am going to make recursive on here or I am going to just fall through and not execute this recursion and stay awake down to this.

So if you say that the partition returns a value which is smaller than the bound is equal to 100 equally over I just return that is irrespective of so many times people confuse between are on the standpoint of expect point. Is pointing to so what we do flu photos remember is that stack pointer of this point screw from there the stack stocks. And activation in the power standpoint our body for efficiency reasons what we will do is that if I look at some activation record my SP will be here and I have an additional 20 bills which sometimes.

We will find exes or the fail pointers and then I can access data within this activation either with respect to stack pointer or with respect to the difference between stack pointer sometimes is that if this activation record. Becomes large then some dates are put the access very quickly by looking at an offset with respect to at P and some data. Which can be accessed with me by move again and offset with respect to S P and I want to keep offset as small as possible so now it should also give you a riddle if you go back now to what I did in my activation record in the data.

Which is frequently accessed is either going to be here or is going to be there so for example if you notice that I had this local data and so this is where I set versioning status in the middle how many do I need pushing status machine . I did not even the coordinates not during the execution of the function but any data is during the execution of the function is kept close to one of the pointer and the reason is that this data is accessed much more frequently there or I should be able to access this more iteration took this data so if this activation becomes large then I want all frequently access data to be close.

To one of the pointers and one of the advantage that I will be able to access the data with a very small offset that means not only I will say by itself in memory but I also have a faster axis jump location or access a location and the offset is not which I think will say outside of 8 bytes and I

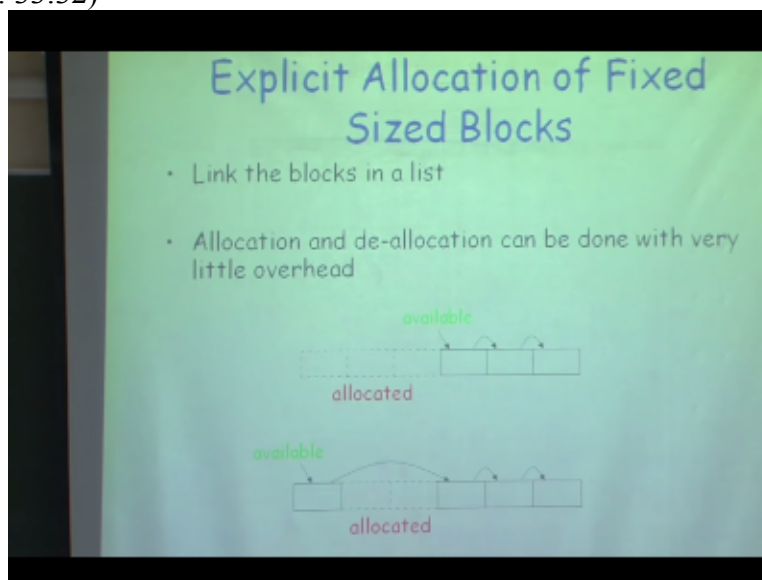
will take would be 3 bits to fit into it it's an offset becomes very large. Then I need to make additional fight who put it off second if this is small instruction and Albany these small offsets are kept the distance and these instructions are much faster so just go back and look at the difference between small offset and offset it stores the whole instruction in fewer bytes. And therefore becomes much faster advances as there is something which are there for us and what we try to do all the time is that we try to exploit whatever hardware gives me so all the so obviously the building we started with spirit 94 and that all the way to go see this in puppets and we see because we know how we had access things. We also know how to create now links move go structure that means are the lexical scoping and I name is an entire clan system so what is the next thing with coordination one thing in common and that is obligated for program indicators. This is my this is so this is so when I say that I have this which is all copy again may have is that we are now in allocated data. Is going be retain of coming what happens is that you can say that give me enough space for creating this kind of data structure there I am saying that v8 mouth record it will determine from the record is to be doing here is what I mean who is that I mean create a mutual sense it is what I will do is I will decide that I want to dispose of this one already is one it now as soon as I eat at this point that none and there is no the pointing to this part of the list. What will happen so they are not part of the free space but they can also be not allocated to any other and I may also create any references so I just say that dispose of this when I say see this particular cell what will happen this pointer is pointing to some arbitrary location this is and in fact if I just dispose it also.
(Refer Slide Time: 28:33)

Language Facility for Dynamic Storage Allocation

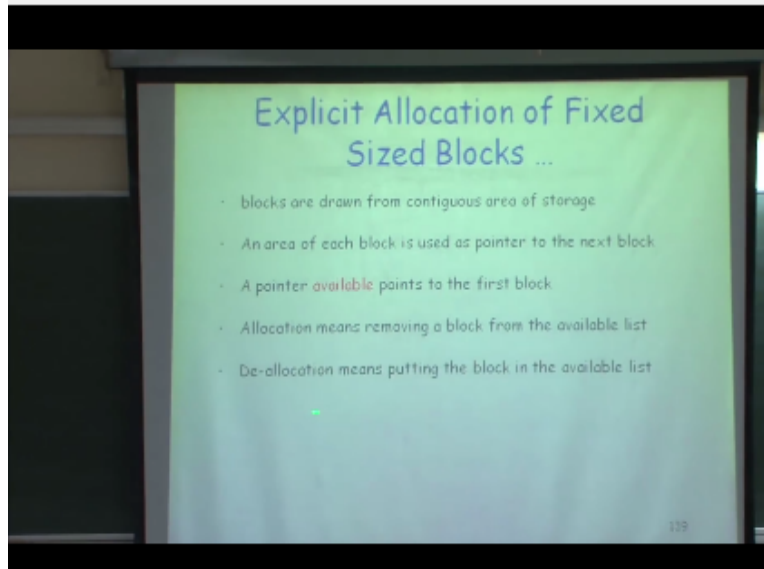
- Storage is usually taken from heap
- Allocated data is retained until deallocated
- Allocation can be either explicit or implicit
 - Pascal : explicit allocation and de-allocation by `new()` and `dispose()`
 - Lisp : implicit allocation when `cons` is used, and de-allocation through garbage collection

So now their language which will manage your storage so that even if you create dangling references and garbage over the period they will clean up the space for you but their languages like C and this classic language is not do any such thing so explicitly program arrest managed part of it by therefore we need what we need to ban it is that where do I store this and how do I manage money compiler like I cannot implement I can enforce program participant but compiler time I need to make sure that whatever is provided by the line together efficiently implement that.

So first thing is explicit allocation of yesterday I gave you a glimpse of this expedition but basically what they are doing is that we have saved that rocks are kept in the list so I am talking about which size box so this is saying that for some small number I can say if I say I locate four hundred fights I said do not worry I will give you all this one piece idle so what we can do is location with very little overhead and what is that all I am saying is that I have an available list. (Refer Slide Time: 33:32)

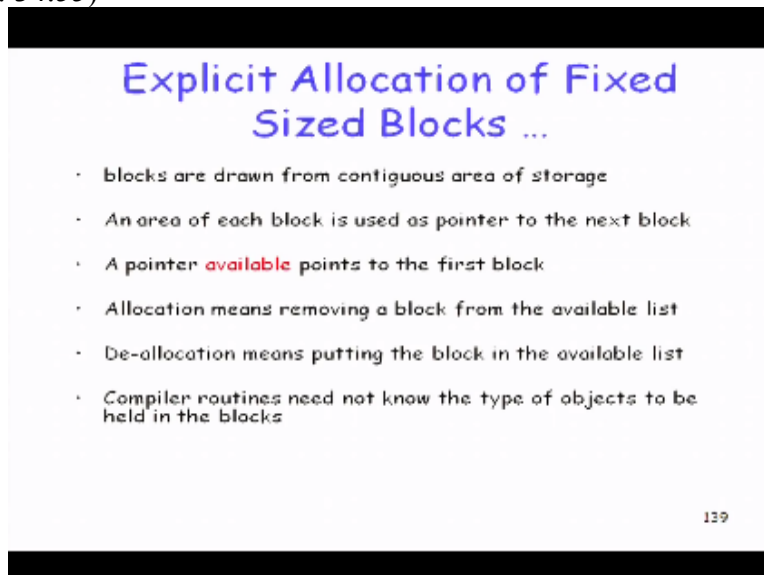


Where I have a linked list of available box to me and then these are the party that we allocated and now if I say about two three this particular cell I want to be allocated to sell this becomes not part of the available list and this part still so you can see that overhead of managing a fixed size lock is very small okay I just need to have a linked list of available blocks and then I can just a location be a location in DL means that I need to put it in the list so this is hardly in your work this really does not put too much of pressure on us and depending on this. (Refer Slide Time: 34:15)



So blocks are going to be drawn from continuously off storage that means as far as this caucus themselves this is three ways to be continuous in maybe I will not say that this law of 1k will take 256 bytes from here in the remaining 780 four bytes somewhere else that kind of thing you not have to do blocks are drawn from and area of each block is used as part of it obviously it is fun to do the next one okay and then I have this point available which points to the first of it will block in this yes and an efficient means to be moving from location means in the back into the variable list okay.

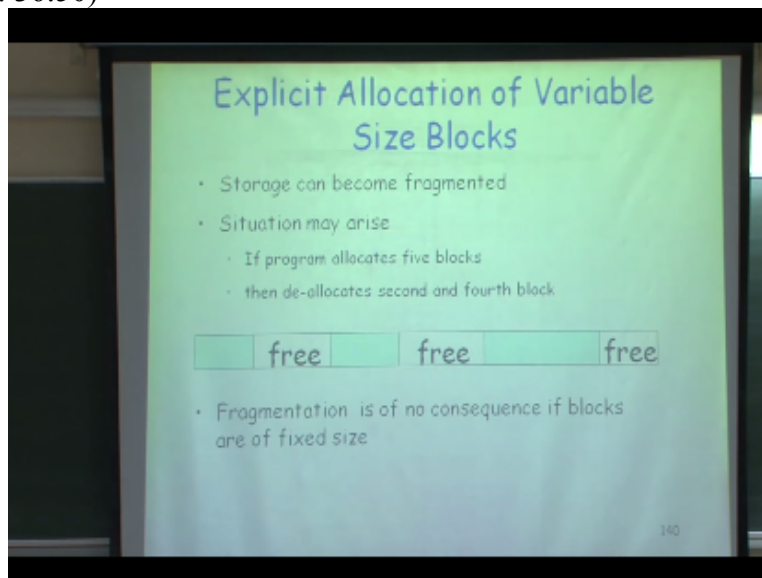
(Refer Slide Time: 34:55)



Now compiler rupees now remember that what I am doing here I am writing part of the program which is going to manage money that is also program but I have to make all the declarations there oh but I allocate space when I say that I have this linked list in what is the type of business

I need so I imagine writing your C program which manages explain this all even is writing the program anyway level back what is the type of the record I declare so normally what we do is as far as runtime system is turns out of the deep manager is concerned it actually needs not know what is the type of this.

Just to have and when I actually allocate it goes to the program programmer is going to determine what type of data they are going to scope so as far as his manager is concerned it manager does not care or in the type of this now interesting situation comes when you have a variable size again you have to worry about best fit or suspect so storage can become fragmented over the years over the time and situation will come and program allocate some box and then you look it is then so I may have this fragmentation going off that I have located somewhere.
(Refer Slide Time: 36:50)



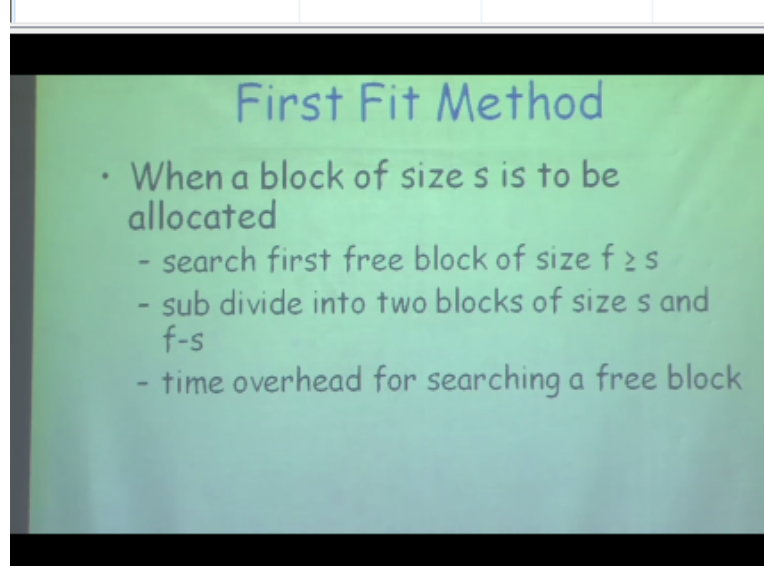
And then the speakers speak and now if I say that I want a block of say size which is larger than all of these low thousands defragment so fragmentation one sequences is lots of sites fix it then it does not matter the pump exciting rock I just became a linked list does not matter where the next point that is that is not an issue but fragmentation becomes very important when it is variable cycles so now I also have to see that how do i if I find that suppose I have a free space here and a free space here and how do I merge two pieces if they are consecutive to each other then merger is straightforward okay.

But if they are not complicated to each other and what I need to do is I need to move this so assuming that this real part is the allocated space I need to move this here and what does movement involved so suppose this is a task in front of you if that I have these so assume that the smallest unit is a fight and I want to make sure that this data remember that once I move in here

and move it here movement just does not mean that I will change the they said this it means something else that I may have pointed it I may have a location which is actually pointing from some other location.

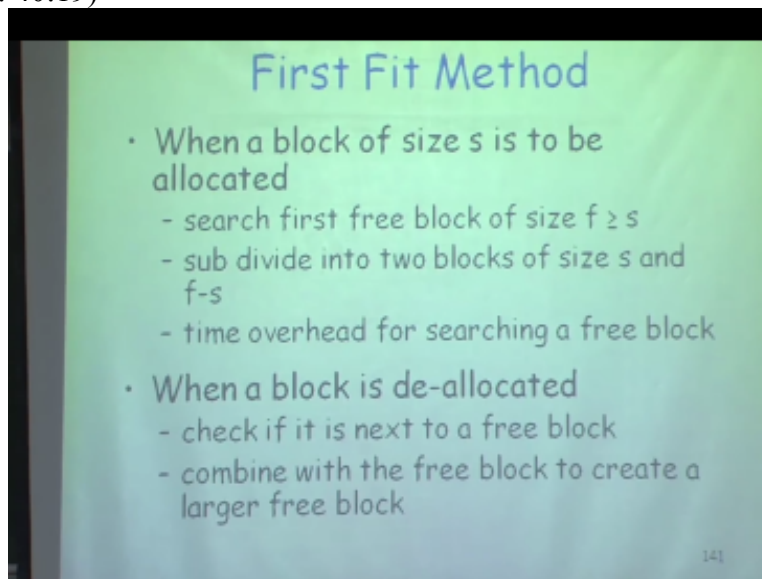
So what I need to know is that if it is pointing to say location X and that X moves now everything which is pointing to X should now start pointing to the new location where X is going so I must know what my pointers are that means when I say that I have some allocated space within this allocated space I should be able to track what my pointers are in water my point when you worry about when I say defragmentation has anyone done ever D transmutations on a hard disk from hard disk.

So how do you think a hard disk defragmentation will manage also suppose I mean you're working on your existence this is your fine system then all the file pointers must be perfect right so all I know structure should not be destroyed in this one so somebody knows in some people that we are all my I know addresses are and when I move this particular block into another block then the in ode table must go back and catch all the information right so similar things will be to here ended shortly purposes. So blocking on the unaffected even if space is available because I mean each table space and because of mechanization so a location of the first method is, (Refer Slide Time: 39:54)



That searched the first two blocks which is of size greater than this and so best fit companies I am looking for a block which is of size just greater than this persons method is that I start looking at this and the other method will speak individual say let us look at the largest block or maybe another method which is saying that after allocation the remaining block should not be smaller than certain size okay.

(Refer Slide Time: 40:19)



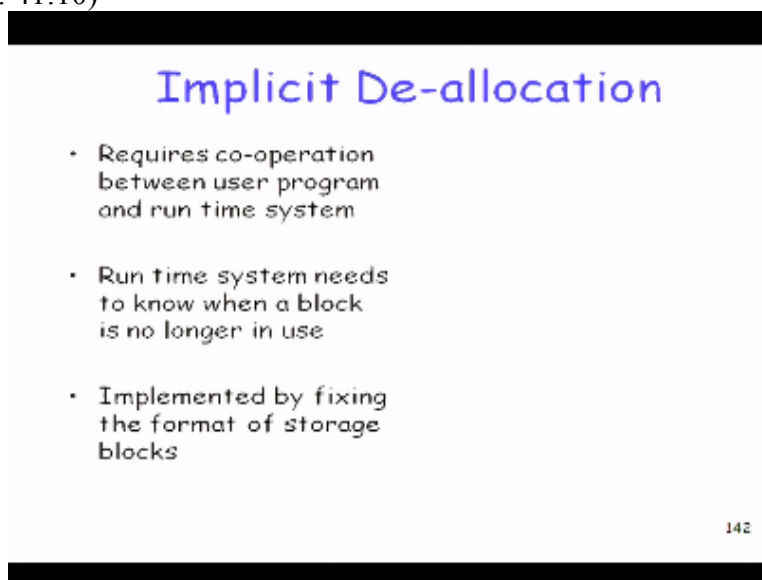
First Fit Method

- When a block of size s is to be allocated
 - search first free block of size $f \geq s$
 - sub divide into two blocks of size s and $f-s$
 - time overhead for searching a free block
- When a block is de-allocated
 - check if it is next to a free block
 - combine with the free block to create a larger free block

141

So on so you can think of various strategies here and then a location means that I look at some block which is of size greater than this and then subdividing and I move ahead obviously there is now overhead of searching for a Pedro so earlier in the fixed size education there was no overhead only overhead was that there was one pointer I was saying that pointer points to the first I just the next one and the block is being located what is done whether this is next block then I just combined it but it is not taxable free block so if this is the education is if you the implicit de-allocation requires now.

(Refer Slide Time: 41:10)



Implicit De-allocation

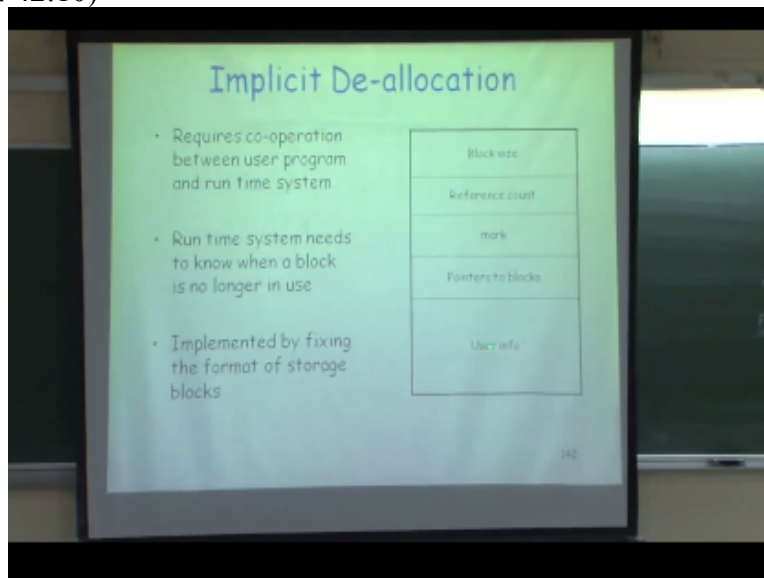
- Requires co-operation between user program and run time system
- Run time system needs to know when a block is no longer in use
- Implemented by fixing the format of storage blocks

142

So when I say please sit in your location and allocation is equivalent to either thiamin collection or over the time or the fragmentation now this requires some cooperation between the user and

what kind of operation is required now I need to know do things I need to know what the clock you have no longer use how do I find out the block which is no longer in use so ok think about it but shortly I am going to ask the description for three and this by implemented by fixing the format of the storage box that means I just cannot say that I get a block and I put some data okay. User program has to make sure or my runtime system has to make sure that the data which is put here has certain structure and what the structure may be it may now have a structure saying that I use information but I also say that what are the point is were the box so when I say user information here.

(Refer Slide Time: 42:10)



It may have some and these which actually are pointing to some other box there are some pointers okay now want to keep these entire pointer separately. So that when I start defragmenting I start a locating I need to worry about what these pointers are pointing to because if I don't have this information I not be able to manage it then I also have certain markers which will say that whether this lock is in use or not and I also have a reference down saying that how many other blocks are pointing to this particular block.

So I need a lot more information than this user information if you want to manage all these be elevations so I have information like block size so this will say that if I have the first address then I know the bounty say I mean think from the point of view of what happens there you just say that you have certain locations and each maybe but if I know that this is the to the location site then I think find out what is the size of this particular one and by having this exciting formation saying that how many rocks whether is in use around how many blocks are pointing to this and what are the rocks it is pointing to that is the one.

Which has me doing a location in which helps me good see a location and defragmentation now once I have this information suppose I have this information this is how I can get it out otherwise we are okay anything of the strategy not that way suppose that I have all these blocks somewhere in my eat which are life and I just need to know only one thing that is the first block in use if I know the first block reduce that you know a pointer to everything so what I can do is I can mark that all by drop stops not in use and then I can say that the first block is in use so I change the marker to that.

And say that this is the news and then I say whatever it is pointing to I who do those blocks and mark them also as used and if I continue this traversal then I will mark all the blocks is used and whatever is not used whatever the marker has not been marked that okay so one way to look at this is that so somebody actually gave a very nice pictorial representation remember seen the video of this I any metadata structure yeah so I mean here I mean if you assume that every pointer is every link is justified you go to the first block and pour some color into this color liquid into this and it is going over.

All these pointers and wherever it does not teach so whatever block is the color that is induce whatever is not color that is no longer reduce and the arrow pimples okay so that means you have lost 30 all those locations in therefore there is no need to keep all this so recognizing block boundaries so now let's look at each of these and the first thing we say is that if talk is of size fix. (Refer Slide Time: 45:17)

Recognizing Block boundaries

- If block size is fixed then position information can be used
- Otherwise keep size information to determine the block boundaries

Whether Block is in Use

- References may occur through a pointer or a sequence of pointers
- Compiler needs to know position of all the pointers in the storage
- Pointers are kept in fixed positions and user area does not contain any pointers

143

Block is a fixed size block then the position information can be used otherwise if sizes because that is I will say that this is where my pockets in case of link less values this was the Pistons hit and when a block is in use is so that means me occur to a point yes or to a sequence of pointers

into vertical pointers and compiler need to know position of all the pointers in the sky and wipers can be kept in a fixed location and in any pointers so basically this part if this, (Refer Slide Time: 45:45)

Implicit De-allocation

- Requires co-operation between user program and run time system
- Run time system needs to know when a block is no longer in use
- Implemented by fixing the format of storage blocks

Block size
Reference count
mark
Pointers to blocks
User info

142

Is where I am saying that I am keeping all the pointers here so basically in this user information I know that for my locations which are pointing to something else which are pointed in. Otherwise I cannot differentiate between a bit pattern whether the address of this wily we explicitly know what my pointers are so that when I start moving other blocks then I know that this is these are the locations which need to be changed so this is where I say that where the block is it use it and, (Refer Slide Time: 46:18)

Reference Count

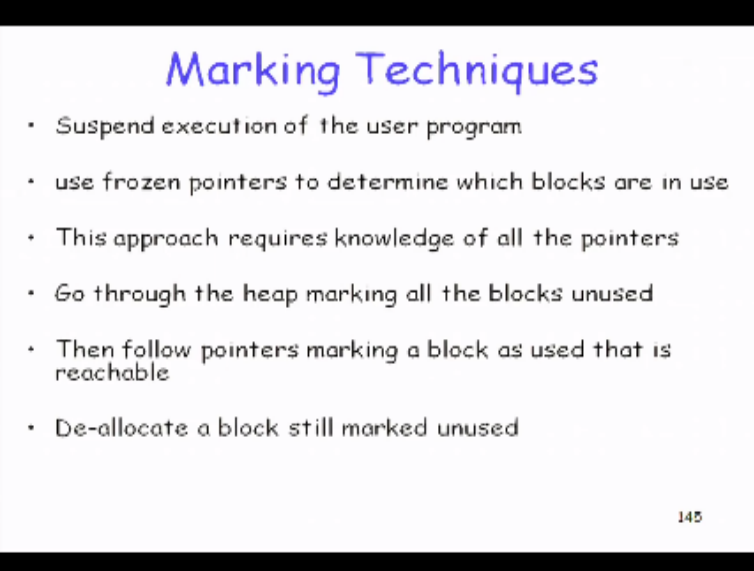
- Keep track of number of blocks which point directly to the present block
- If count drops to 0 then block can be de-allocated
- Maintaining reference count is costly
 - assignment $p=q$ leads to change in the reference counts of the blocks pointed to by both p and q
- Reference counts are used when pointers do not appear in cycles

144

What is the reference outfit each pack of number of blocks which point directly to present every time some point that is disposed in one of those blocks now I need to come back here and change my reference compare and whenever this reference now becomes zero. Then I know that this is

not being fun to buy any other block and if block to zero then can be allocated and because every time I do like this it says Q is easy sign Q then what is happening here that whatever q was pointing to so if you B is a pointer whatever it was pointing to it will so technique use so marking use.

(Refer Slide Time: 47:49)



Marking Techniques

- Suspend execution of the user program
- use frozen pointers to determine which blocks are in use
- This approach requires knowledge of all the pointers
- Go through the heap marking all the blocks unused
- Then follow pointers marking a block as used that is reachable
- De-allocate a block still marked unused

145

Technique is now this automatic be a rotational applause. So what we have to do is now suppose your program is you do and you say that I want to find out whether certain blocks are reachable or not. Yes then if two processes are running simultaneously this can become nightmare because I am continuously allocating and the allocating and the other process is trying to find out that is trying to find out whether certain block is in use or not so this cannot go on the first thing that practices that you have suspended in fact if any one of you have done this programming or have done programming using some kind of interactive development environment of a language. Where garbage collection happens then you will notice that you editing your program the program is executing in everything stops and then you for one minute you are just wait nothing happens so basically it is the garbage collector which is running in the background and your process can be which has being suspended because if you want to find out one of the pointers in use you cannot simultaneously modify your pointers and say find out what are the quantities. So suspend execution of the program and then whatever pointers are close in pointless that you be dumping what are the rocks which I going to use and this is the approach which requires knowledge of all the pointers but we have seen the structure with all the pointers are to be kept separately now the heat waves start marking all the blocks which are unused and once we have done that then comma pointers which mark the block is used and that is reachable so first block

will always be reachable I am asked that is used and then be a locator clock which is still not right.

So I start with the first lock the first use clock and say that take pointers from here and continue marking and compaction is going to move all the use of and of E so this is defragmentation ISM and pointers all the pointers in objects at district okay since I know the locations of all the pointers now and what it is pointing to that movement will now become easier not have to worry about so if you know your layout of the space that what my pointer something I can do it right so this is where we'll close about the discussion on put management of e and we without get into the Action Coalition which is looking at the final machine code okay and that we solved.

Prof. Phalguni Gupta

Co-ordinator, NPTEL IIT Kanpur

Satyaki Roy

Co Co-ordinator, NPTEL IIT Kanpur

Camera

Ram Chandra

Dilip Tripathi

Padam Shukla

Manoj Shrivastava

Sanjay Mishra

Editing

Ashish Singh

Badal Pradhan

Tapobrata Das

Shuubham Rawat

Shikha Gupta

Pradeep Kumar

K.K Mishra

Jai Singh

Sweety Kanaujia

Aradhana Singh

Sweta

Preeti Sachan

Ashutosh Gairola

Dilip Katiyar

Ashutosh Kumar

Light& Sound

Sharwan

Hari Ram

Production Crew

Bhadra Rao

Puneet Kumar Bajpai

Priyanka Singh

Office

Lalty Dutta
Ajay Kanaujia
Shivendra Kumar Tiwari
Saurabh Shukla
Direction
Sanjay Pal
Production Manager
Bharat Lals
an IIT Kanpur Production

@Copyright reserved