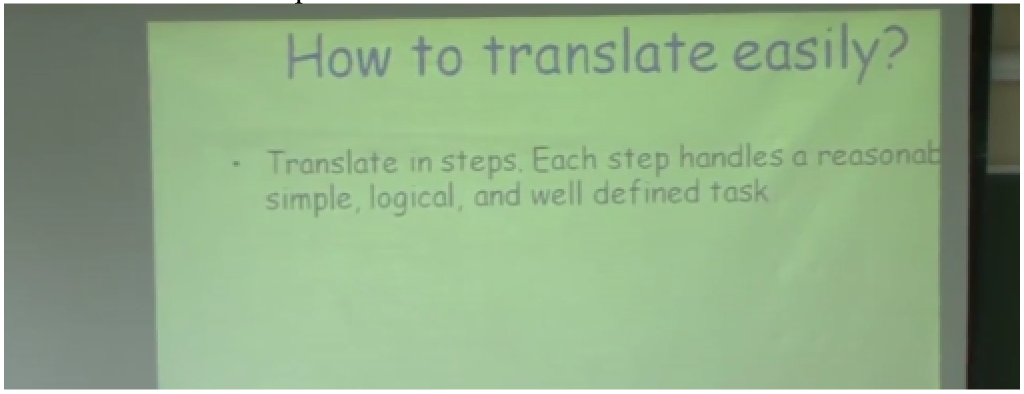
Indian Institute of Technology Kanpur
NP-TEL
National Programme
On
Technology Enhanced Learning
<u>Course Title</u>

Compiler Design
Lecture – 02

By
Prof. S.K Aggarwal.
Dept. of Computer Science and Engineering

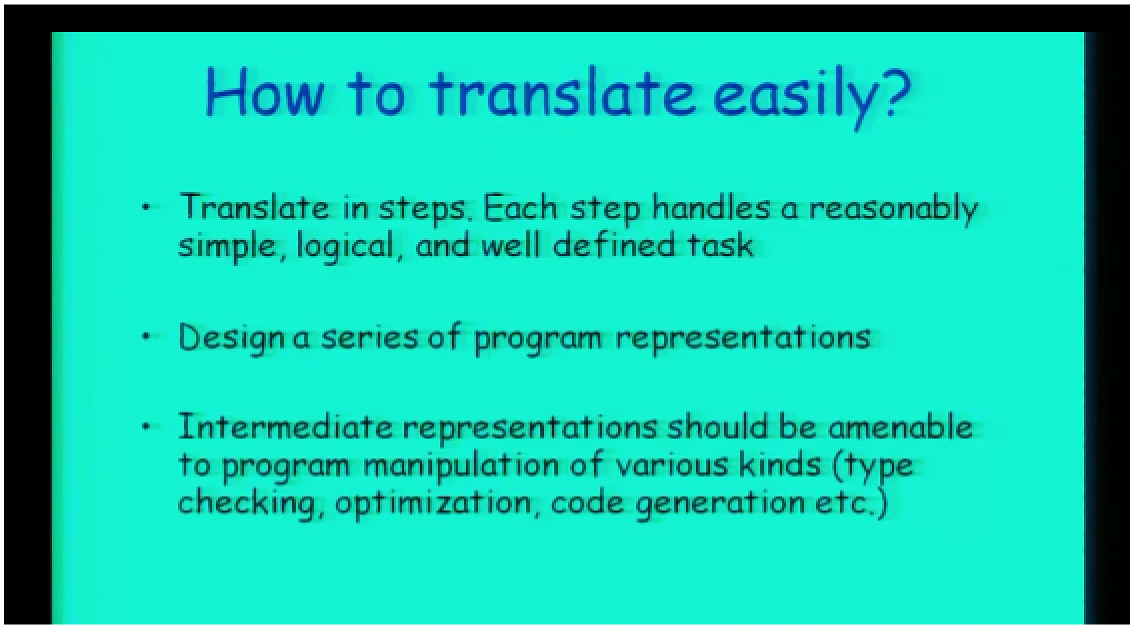
Good morning. So let us start discussion for today and quickly to summarize what we started looking towards end of this class, we were looking at all compiler fits into the overall programming development environment and what compilers does, and how compiled does, okay, and we were really started the discussion on how compilers works? And we were saying that if I have to jump from a representation which is in a high-level language to a representations which is a low-level language and compiler is sitting somewhere in between, then we don't want to do translation in one step, but we want to take small steps at a point of time and we want to move these small steps, and we need to understand, to understand compiler what these small steps are, and if we can figure out each of these small steps then perhaps we'll be able to understand how compiler works.



How to translate easily?

- Translate in steps. Each step handles a reasonab
simple, logical, and well defined task

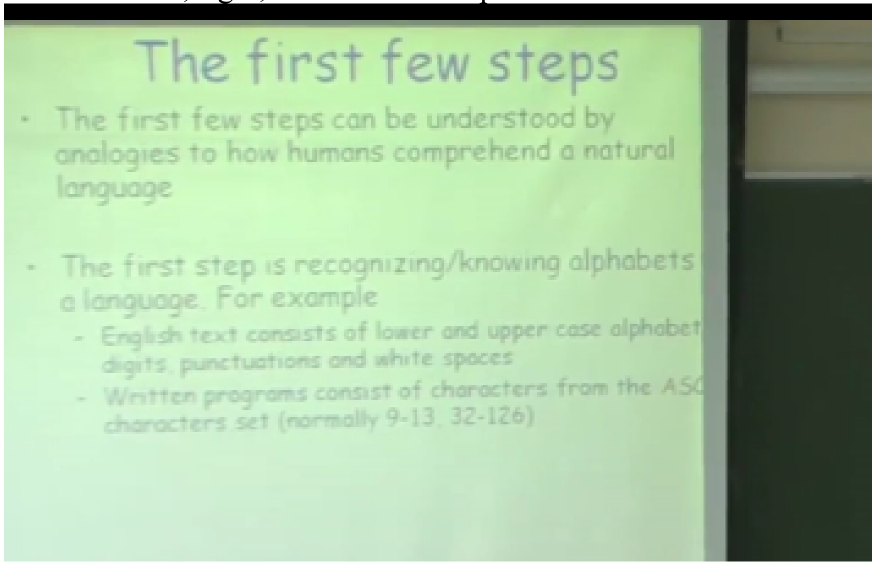
So you want to translate in two steps, we want to make sure that each step we are talking about here is doing some activity which is coherent and which is logically isolated to one of the phases that is going to help us in development, that is going to help us in final debug, okay.
So what we want to do now is, we want to design a series of representation, as I said that what we are doing is really changing the representation from high-level language to representation to low-level language, okay and therefore if I need to have these small steps I must have a representation here, I must have a representation at each of the steps, and slowly what I am doing is I am just converting this representation till I reach the target here.



How to translate easily?

- Translate in steps. Each step handles a reasonably simple, logical, and well defined task
- Design a series of program representations
- Intermediate representations should be amenable to program manipulation of various kinds (type checking, optimization, code generation etc.)

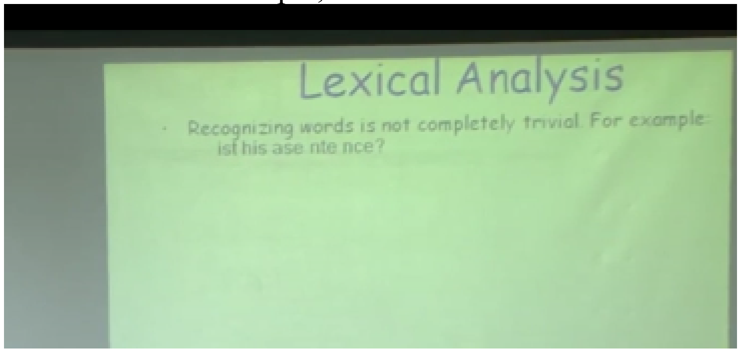
So what are these steps and we started looking at few things, okay, so we are looking at intermediate representations, now kind of so what we call these, are intermediate representation, this is the one initial representation, this is the final representation, and all these are intermediate representations, and we need to understand what these intermediate representations are and how each of these phases is going to take one representation and then convert that into another representation.
So as I start from this place you can see that this representation is very close to the higher-level language, and this representation is close to the machine language or is the machine language and as I move above this path, representation keep becoming closer and closer to the machine language, that is our idea, and the first step we looked at was that we want to understand what this representation is, we want to find out what are the alphabets in my set, and I want to find out various words and so on, right, so we had examples of that.



The first few steps

- The first few steps can be understood by analogies to how humans comprehend a natural language
- The first step is recognizing/knowing alphabets a language. For example
  - English text consists of lower and upper case alphabet digits, punctuations and white spaces
  - Written programs consist of characters from the ASC characters set (normally 9-13, 32-126)

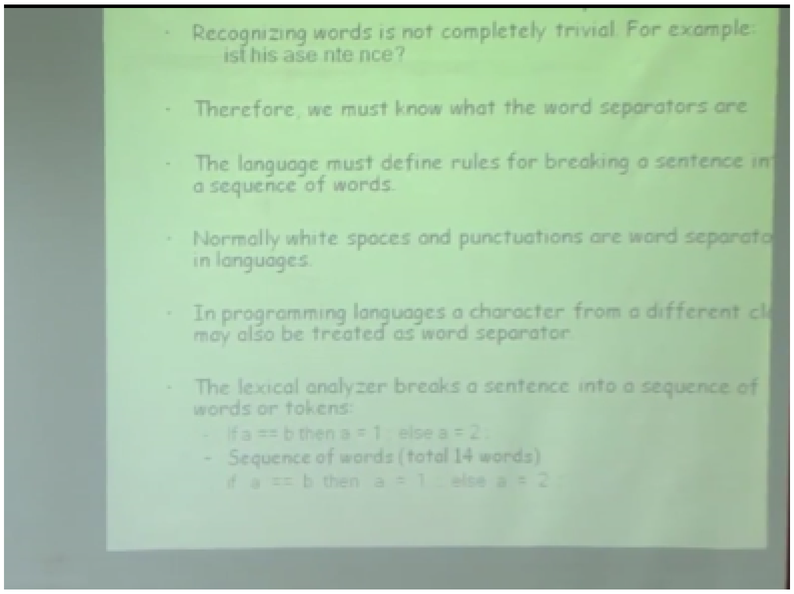
So first few steps can really be understood by analogy how we understand the language, and that is the example I was giving you, so what we want to do is that we want to first define what is the alphabet set I have in my line, okay, so if I am trying to make a compiler for any language, first thing must be known to me is set of alphabets, okay, this is the, so we had example of English, but if I talk of, see what is my set of alphabets, you can see as the characters then, all the visibility characters, okay, you don't care about control characters. And once I have defined a set of alphabets then I look at set of words, okay. And what are my sets of words in the typical programming language? This could be keywords, and apart from keywords what else would be there? Identifiers, then I have operators, I have all the, so called punctuations, so when I say I have semicolon so on, okay, I can have numbers, numbers will come in various representations and so on, okay. So I'm going to have all these set of words and I must also know how to break my set of alphabets into a set of words, okay. So this is what we're looking at, so here is another example,

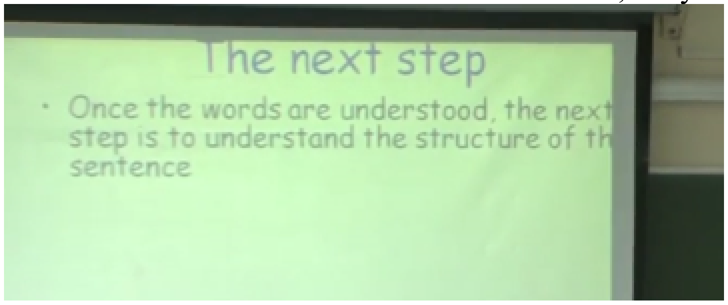


Lexical Analysis

- Recognizing words is not completely trivial. For example:
  ist his ase nte nce?

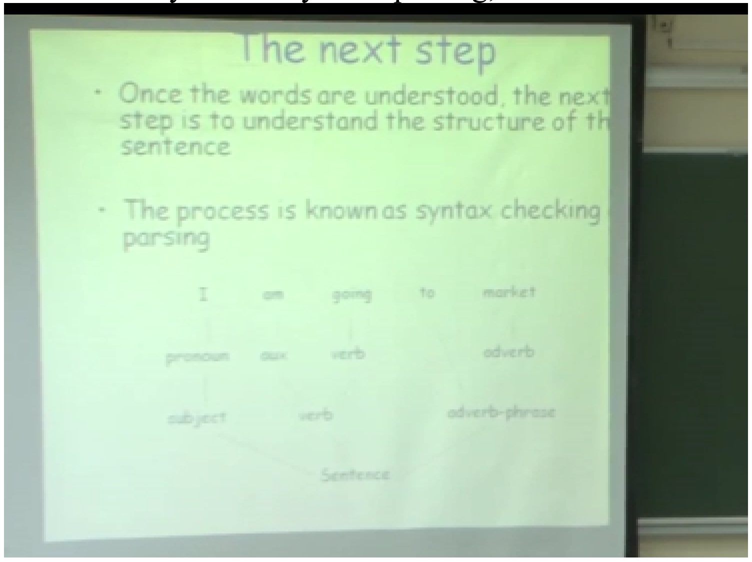
okay, now it may sound that I mean it's very easy to find out words but look at a sentence like this, how many words does it have? What is the sentence? How do you know? So you can see that normally I have violated all the rules I know of English, but you are able to figure it out, alright, so you can go back and forth and find out that well this must be word boundary, is this a sentence and I have put blanks at all the wrong places, but you're still able to figure it out, okay, but compilers can't do that, so compilers must have very well-defined set of rules which will say that how to breaks a sequence of characters into a sequence of words, okay, and this is the first phase which is known as the lexical analysis. So this is really the first phase of compilation, let me say that input is, so let me just use the same figure, so if I say the first phase, I'm talking about is the lexical analysis, okay, and lexical analysis is the one where input is set of characters and outcome here is a set of or sequence of words, okay.
So this is the first thing I want to do, given something not as difficult as this, but something much more trivial so if I say I'm writing a statement like this, I'm writing an expression like this, the first thing I want to do is I want to break this into words and say what are my word boundaries, I'll identify all the word boundaries and ignoring anything which does not want to move to the program, so if they're comments I'll ignore them, they're blacks I'll ignore them, because they are just for indentation, they're not contributing anything to the meaning of the books, okay, and therefore we must know exactly we have got the word separate our self, right now words separators, depending on the language should be different, so in English you say that whenever you have a space, whenever you have a tab and so on you have a word separator, so the language must define what are the rules I must have to break a sequence of characters into a sequence of words and normally what we have are white spaces, punctuations, etcetera these are really the separators, okay.
And in programming languages characters form different blocks can also be featured as word separator, so for example I may not have a punctuation here, I may not have another word here, so you can see that sometimes even when I don't have a blank for punctuation, I'm saying this is

a word boundary because this is a word which is coming or this is a character which is coming from a different blocks, okay. So for example if I have a sentence like this or a sequence of characters like this, which says if A=B, then A is sign 1, as A is sign 2, okay, then sequence of words will be that I will break this and say that here is a break, here is another break, here is another break and so on, okay, and I know the rules how I am going to break this and ultimately what my lexical analyzer is going to do is, it's going to take this as input and it's going to this, this is all, and this is what, this is really the first step of what a compiler does, make sense, okay. Now once I have broken this into a sequence of words, what should I do? What does tagging mean?

- Recognizing words is not completely trivial. For example:
  isthis ase nte nce?
- Therefore, we must know what the word separators are
- The language must define rules for breaking a sentence in
  a sequence of words.
- Normally white spaces and punctuations are word separato
  in languages.
- In programming languages a character from a different cl
  may also be treated as word separator.
- The lexical analyzer breaks a sentence into a sequence of words or tokens:
  - If a == b then a = 1 ; else a = 2 ;
  - Sequence of words (total 14 words)
    if a == b then a = 1 ; else a = 2 ;

So there is, it's not called tagging, what it is called is that I want to really identify the structure and I want to find that with respect to a set of specifications which we call in the languages as grammar, we want to find out whether this is a valid sentence or not, okay.

The next step

- Once the words are understood, the next
  step is to understand the structure of th
  sentence

So the next step really is that we are trying to now once we have understood each word, we want to understand the structure of the sentence okay, so for example okay, let me just pick up the previous, from the previous class. Now here is a sentence, okay, now is this a valid sentence, grammatically, okay. What about this? So, why this is not valid? What is wrong with this sentence? So some of you say that, so let's also look at context, okay, and understand the context clearly, what I'm trying to do now is I'm trying to find out whether this is structurally a valid sentence, and how do I define my structure? I define my structure by saying that my structure or a sentence will have typically if you remember your Wren & Martin will say that there is a subject, there is a verb, there is an object, but it doesn't say to begin with when I talk of structure with what kind of subject, what kind of verb they use, only structure, right, as far as structure is concerned this is valid and this is also valid, there's a verb, alright, I'm not looking at relationship between that okay, so for example if I say structurally this is also correct, okay, there's a verb. Now what will happen is that, this is not the right verb as far as this kind of I is concerned, right, but structurally is fine, so what we are doing here is we are just testing the structure and not looking at the context, okay, context will come somewhat later, okay, so this is the process which is known as syntax analysis or parsing, and if I have a



The next step

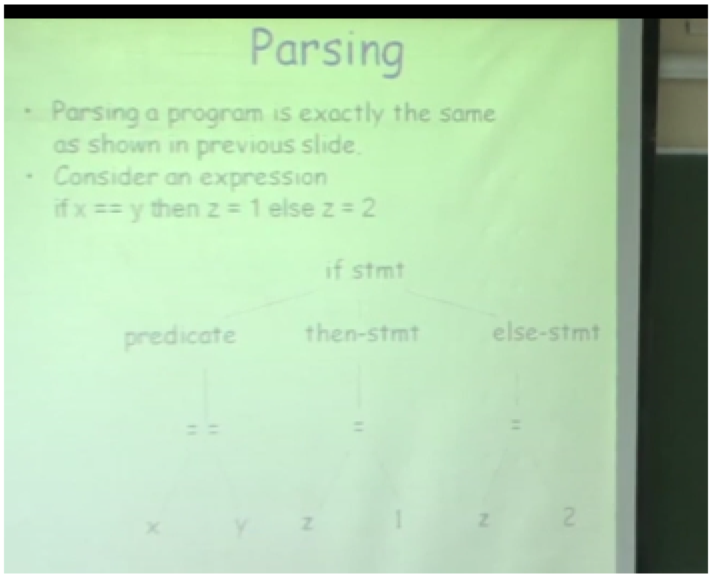
- Once the words are understood, the next step is to understand the structure of th sentence
- The process is known as syntax checking parsing

I
am
going
to
market
pronoun
aux
verb
adverb
subject
verb
adverb-phrase
Sentence

sentence like this I'm going to mark it, okay, it'll give me a structure like this, which will say that the sentence must have subject verb and adverb phrase, and could be auxiliary verb and so on, okay, and this gives me a sentence, but if I put something else here then that will also verb, that is also a valid sentence as far as with respect to the grammar of the language, okay with respect to the structure, okay and therefore you have to be careful when you are thinking about saying that what is valid and what is invalid, okay so for example is this structurally a valid sentence.

So typical my specification could be, I can say that whenever you have an assignment, on the left hand side you must have a variable, on the right hand side you must have an expression, but I don't say whether this is a valid expression or not, okay, that becomes subsequent, okay. So the second phase therefore becomes context analysis where input is of sequence of words and what is the output? So output will be something like this, and this is known as parse tree but it will also tell me whether this parse trees a valid parse tree or not, so for example if I have something like this, that this will not come for that parse, this will say that whatever we are trying to parse is incorrect so if it is, input is correct it will give me a parse tree, if input is incorrect it'll give me an error, that will flag we have the errors.

Now once we have done this, okay so parsing is, program is exactly going to be same as we saw in the previous case of English so if I now try to say that there is an expression which I want to parse, okay, so this is saying if X = Y then value the same 1, else that is signed 2, okay this is going to be my specification which says that if statement must have a benefit, must have a then-part in the else part, and this is how the predicate and then else to, okay, so if this is not valid I'll not be able to construct parse tree and I'll have to flag an error there, okay, so this is what we know as parsing.

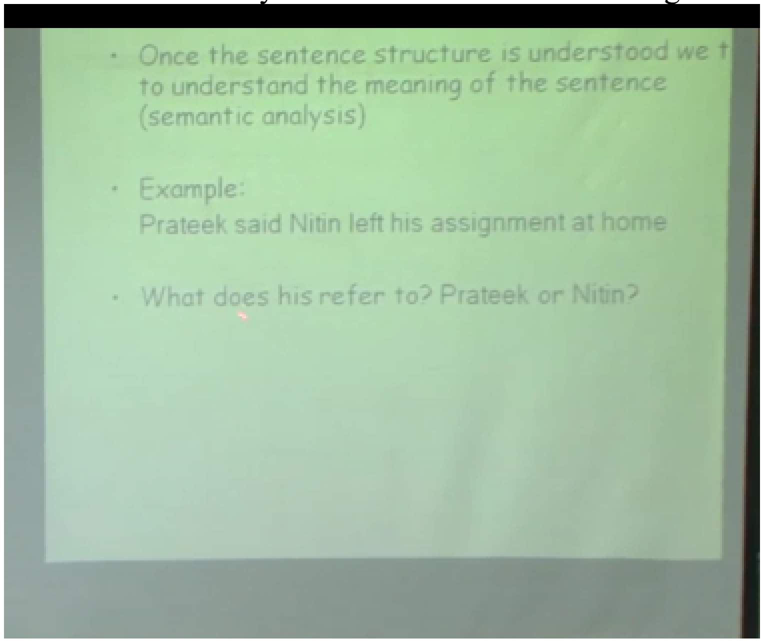


Parsing

- Parsing a program is exactly the same as shown in previous slide.
- Consider an expression
  if x == y then z = 1 else z = 2

if stmt
predicate
then-stmt
else-stmt
= =
=
=
x
y
z
1
z
2

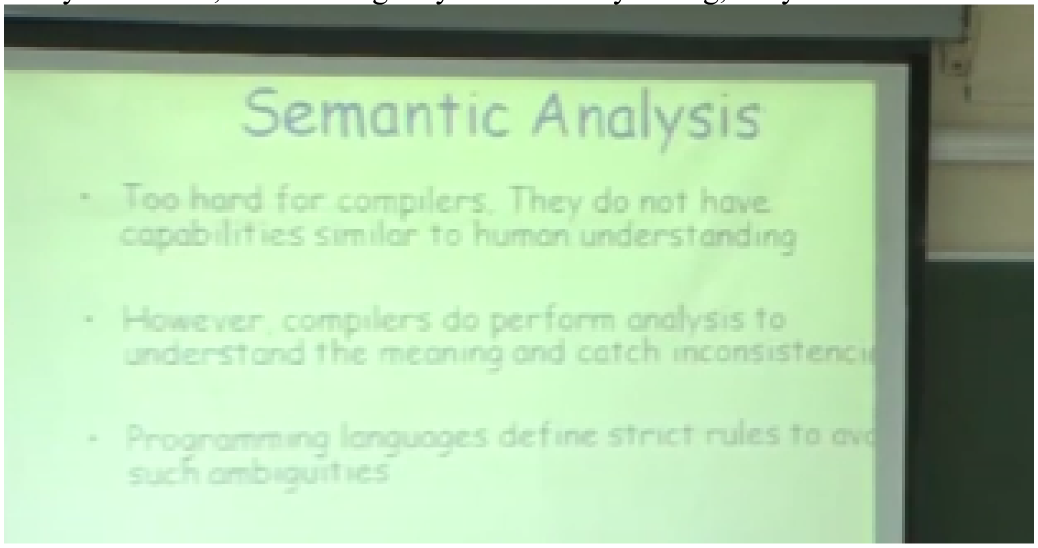
Now beyond this point, we want to understand the meaning, okay, so this is where the meaning starts coming in, and we say that if I use the sentence like B's in C as 35, you said, this just doesn’t make sense, because I’m not using the right kind of sub phrases to find the meaning of this, so this is also known as semantic analysis of program, where I'm trying to understand meaning of the program, so here is an example and natural languages the reason I am picking from natural languages is that they can be quite complex and obviously programming languages are going to be lot more easier but they become easier because we over specify them, we try to put more specification. So when we say Prateek said Nitin left his assignment at home, okay.



- Once the sentence structure is understood we t to understand the meaning of the sentence (semantic analysis)
- Example:
  Prateek said Nitin left his assignment at home
- What does his refer to? Prateek or Nitin?

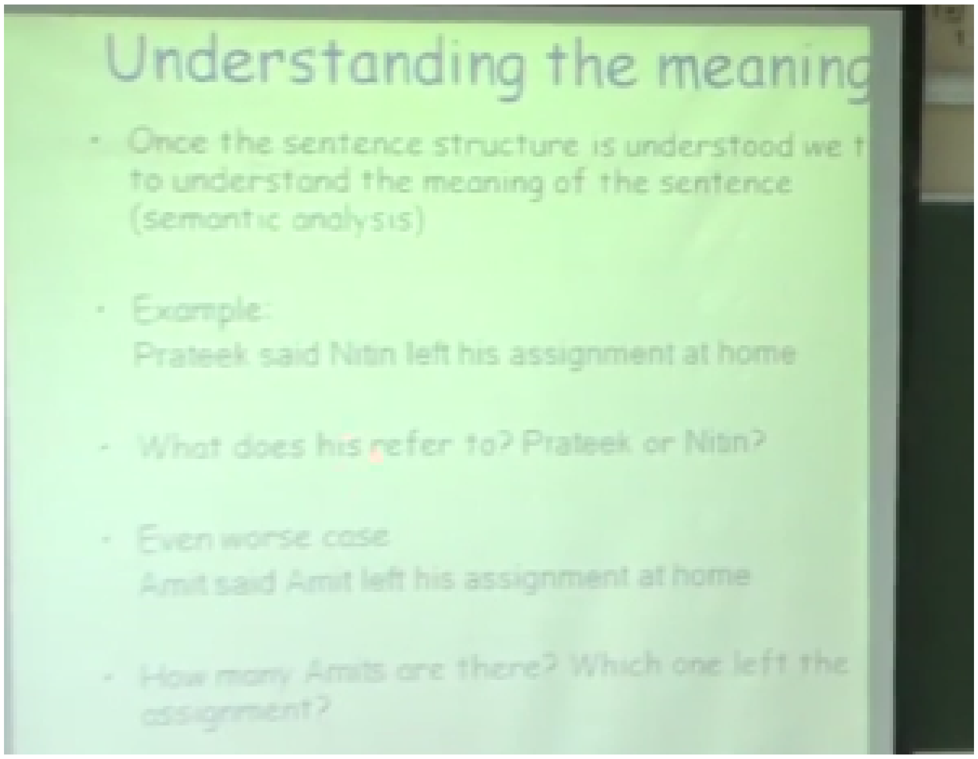
Now you're saying that so look at this scenario, okay, his assignment, whom he is referring to, okay, people can come up with multiple interpretations, okay, now we don't want such a situation in programming languages, we want to make sure that when I write a program it has only one meaning they cannot be multiple interpretations of program, okay and therefore we have to be very precise, okay and there is a worse case than the previous one, okay, when we

say Amit said Amit left his assignment at home, okay, maybe they might be there two persons with the same name.
So how do I handle semantic analysis? Give me some ideas? You have already done this in some way, so let me go back to the same here, is this valid sentence in C, semantically, yes, no? Everyone say yes, wonderful, okay, so let's say, A is of type character, and B is of type integer and C is type float, is this valid? So what has gone wrong suddenly? So how could you say A assignee B + C is valid without looking at correct information? So answer should have been when I say is this a valid sentence in C, you say I don't know because I don't have the context information, like here if I say is this the verb, just by looking at this verb you cannot say whether this is right unless you look at the context, and in programming languages the context is provided by the type information, so I have declarations in the beginning and if I say that I declare type of each and every variable and then you have rules which say that what kind of type variables can be put together in the expression, okay if I try to put wrong kind of type variables so for example maybe this assignment is not even defined, okay, so if I take this addition maybe I mean if I try to add integer and floating point numbers internally it will say let's convert everything to floating point number, but if I say that assign now a floating point number to a character, you say this is an error, alright. So without having the context which is this information I'll not be able to figure out whether something is valid sentence or not, so structure may be correct, but meaning may be absolutely wrong, okay.


Semantic Analysis

- Too hard for compilers. They do not have capabilities similar to human understanding
- However, compilers do perform analysis to understand the meaning and catch inconsistenci
- Programming languages define strict rules to avo such ambiguities

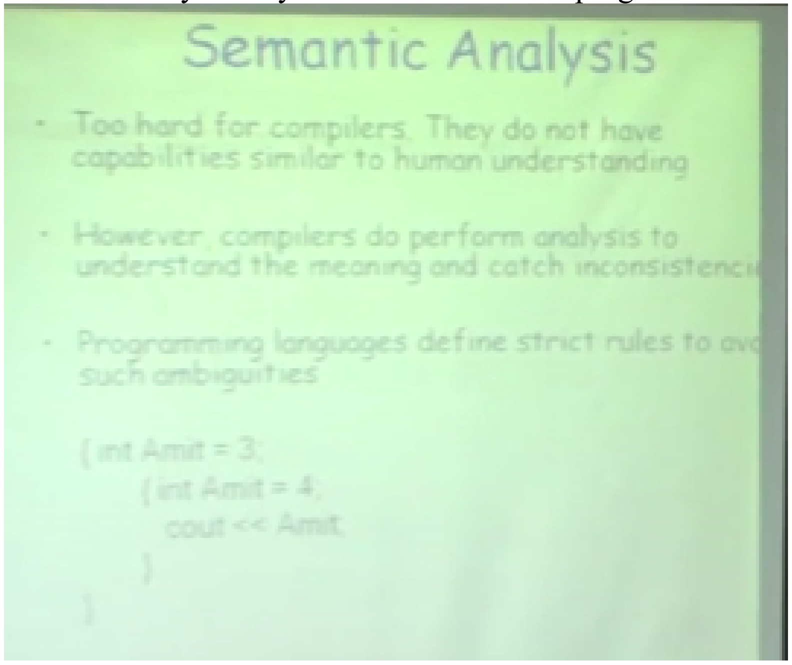
So compilers performed this analysis to understand the meaning and we catch inconsistencies and we have very strict rules to avoid such ambiguities. So for example, when I was having



Understanding the meaning

- Once the sentence structure is understood we t
  to understand the meaning of the sentence
  (semantic analysis)
- Example:
  Prateek said Nitin left his assignment at home
- What does his refer to? Prateek or Nitin?
- Even worse case
  Amit said Amit left his assignment at home
- How many Amits are there? Which one left the
  assignment?

these kind of sentences, okay there ambiguities here which will not permit in programming languages, so I will immediately if I say that I want to write a program like this and I say output

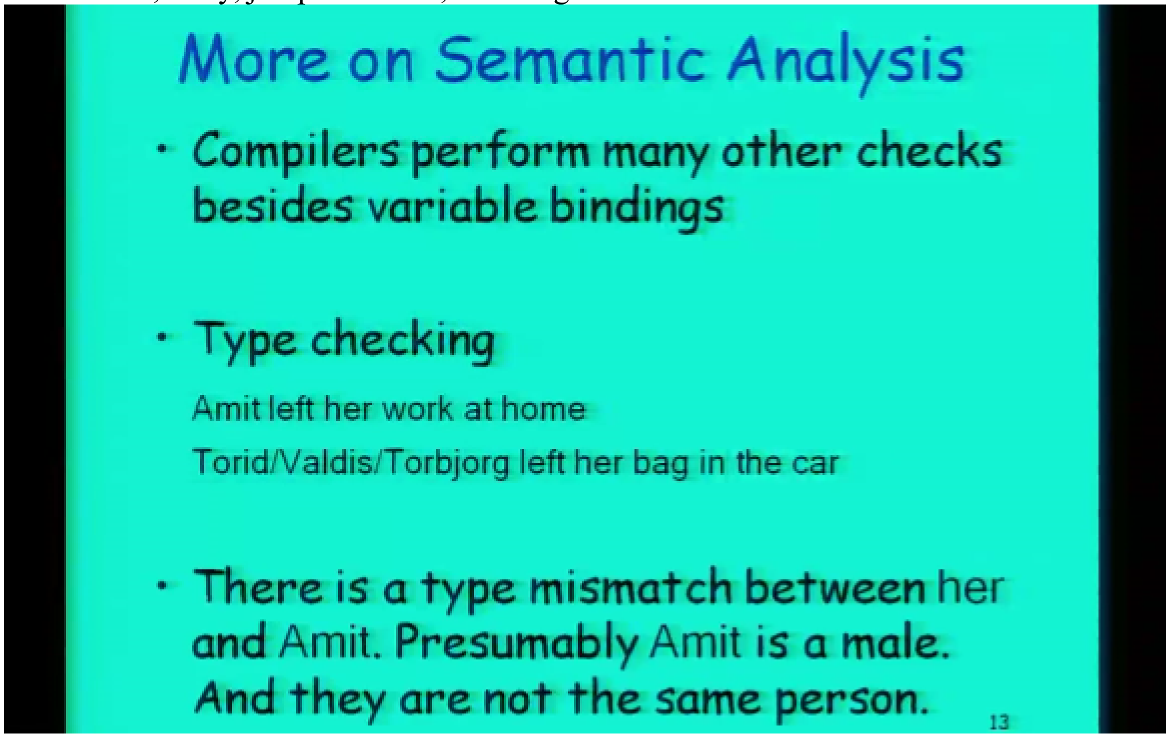


Semantic Analysis

- Too hard for compilers. They do not have
  capabilities similar to human understanding
- However, compilers do perform analysis to
  understand the meaning and catch inconsistenci
- Programming languages define strict rules to av
  such ambiguities

```
{ int Amit = 3;
     { int Amit = 4;
        cout << Amit;
     }
}
```

Amit, then losing scoping information I will know that in this particular scope this variable Amit is assign value 4, there's no ambiguity there, okay, but if I put a C out after this okay then I know from block structure that the value and output is going to be 3, okay, so precisely I know at each point of time in the program that if I refer to a variable what type I am using and therefore what is the binding of this variable, okay, and that is very essential for us, okay, so compilers perform many other checks other than just finding these variables to many types and what are these, sorry, jump far ahead, we can go back there.



More on Semantic Analysis

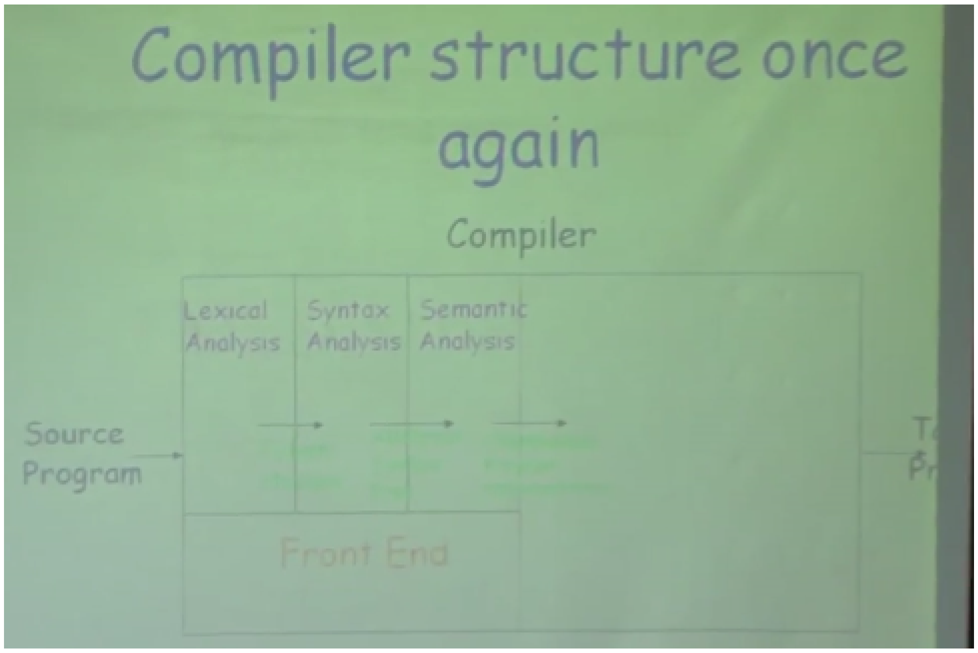
- Compilers perform many other checks besides variable bindings
- Type checking

  Amit left her work at home

  Torid/Valdis/Torbjorg left her bag in the car
- There is a type mismatch between her and Amit. Presumably Amit is a male. And they are not the same person.

13

So here is type checking where we are looking at context information, okay so if I write a sentence like this, okay what does it mean? Do I have all the type information? So you say Amit left her work at home, is this a correct sentence? Why do you say it is not correct? Some people say this is correct, some people say not correct. Okay, and so that is one possible error that Amit maybe referring to, we don't know the context and there may be other sentences, but suppose I say this is standalone sentence, there is no other context then is this valid? Why? You are something here. Okay, and what about this, so these are Scandinavian names, so from your social context we knew that Amit is normally a male's name, not be a female name, now do you know the social context here? Can’t figure out, right? So these are, I just picked up some names from dimensions, features dictionary I don't, so this context is very important for us, okay, and unless we know the context you'll not be able to find out the meaning of the sentence, okay.
So here is some context information but we want stronger context information and what we do here is therefore we have all this declaration and therefore now how does my, at this point of time how does my compiler structure look, okay we have a lexical analysis phase which is going to take this representation and convert this into a sequence of words and then we have and this is also known as stream of tokens. And then we have syntax analysis phase, syntax analysis is going to take input as sequence of programs and is going to give me a syntax tree. And then I have semantic analysis, which is going to then find out the absolute meaning of the sentence which will give me an unambiguous program representation, where I'll have no ambiguity left as far as meaning is concerned, and this is also traditionally known as the front end phases of the compiler, okay, and why they’re called content because typically historically we'll go into little more details, slightly later, what this means is that I am really dealing with the source programming language here,

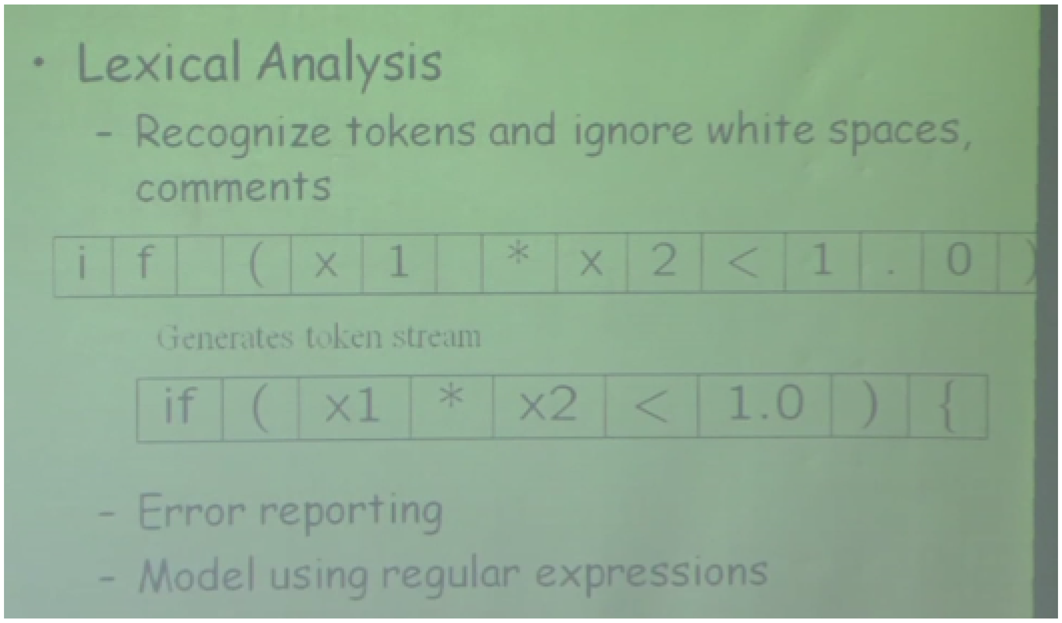


Compiler structure once again

okay, you can see that when I did all this analysis, when I was trying to do all this analysis, I had no information about the machine, I did not know where this program is going to be compiled, where this program is going to get executed, only information I had was of the source program representation and the source programming language, therefore this also is known as the front end.

So this is what we do through the analysis of the program to find out what is the precise meaning of the program, and if I cannot find a precise meaning, then what do I do? I'll flag an error, I say that this program is not correctly written with respect to the specifications of the language, program they still have bugs, program may have logical errors, but as far as language is concerned it is very fine, so I am not trying to find out whether it has logical errors or not but I'm trying to find out is weather with respect to the specifications of the language, I have a correct program or not, clear? Everyone is with me up to this point, okay. So let's move forward because we are still somewhere in the middle and this part still needs to be filled, okay.

Now what are the other things I can do, okay, so do you have front-end phases, let's go back and now look at front end phases and little more details with respect to the programming languages, so we have lexical analysis which is going to recognize tokens, and when I recognize tokens, I also want to ignore all the information which is redundant to the understanding of the program, so for example if you put that somewhere, put multiple spaces, put comments, I just want to ignore that, and if this is my input which is sequence of characters I want to generate the sequence of tokens here, so here is something which is known as a token or a word, okay and from the compiler terminology we keep using word token, we'll not use the words, okay,



- Lexical Analysis
  - Recognize tokens and ignore white spaces, comments

i	f		(	x	1		*	x	2	<	1	.	0	)

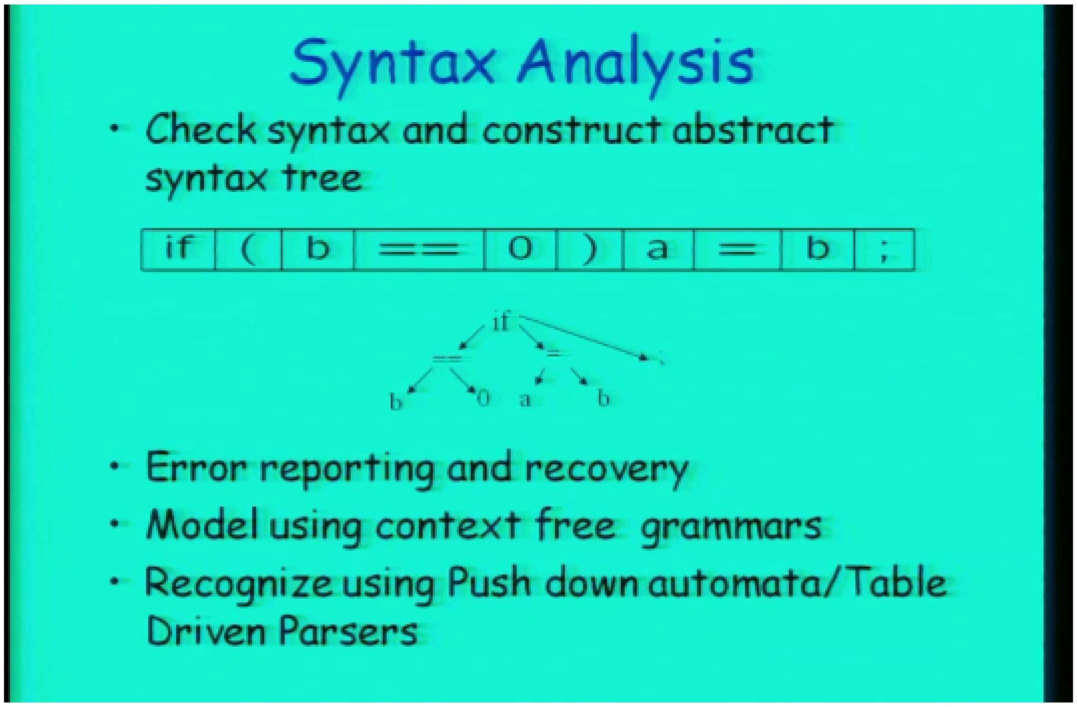
Generates token stream

if	(	x1	*	x2	<	1.0	)	{

  - Error reporting
  - Model using regular expressions

so this is my input to lexical analyzer and this is my output of lexical analyzer, and this really becomes the specification, so when we start doing lexical analysis in detail that is the specification we are going to use, okay, and we also want to report errors if they are incorrectly form words, then I want to report so if you say there is no valid identifier or I am not sticking to the rules of identifier then I report that as an error, and all this will be modeled using regular expressions.

Remember I was talking about how TOC is going to be used here, theory of computation, so use regular expressions and how do I implement regular expressions? I try to make a finite state machine, yeah Finite State Automata, so I'll create a Finite State Automata which will take this as an input and will give me this all, so this is really if you look at this foil it gives you full specifications of what lexical analyzer is supposed to do, what we need to understand it, how it does, clear? Okay, let's look at specifications of syntax analysis, so I want to now check syntax or structure of the language so what will be my input? Input is going to be a sequence of tokens, and what will be output? Output will be a parse tree like this, output you say that all this can be put in a structure line, this, okay, but if it cannot be put in a structure then it will also flag an error, so it will report an error and we will try to recover from this error, what does recovery mean here?



Syntax Analysis

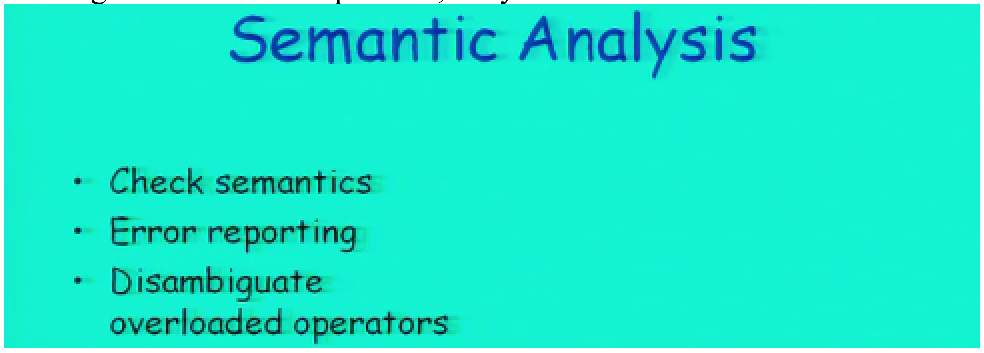
- Check syntax and construct abstract syntax tree

if	(	b	==	0	)	a	=	b	;

$$\begin{tikzcd}
 & & \text{if} \arrow[dl] \arrow[d] \arrow[dr] & \\
 & == \arrow[dl] \arrow[d] & = \arrow[d] \arrow[dr] & ; \\
b & 0 & a & b
\end{tikzcd}$$

- Error reporting and recovery
- Model using context free grammars
- Recognize using Push down automata/Table Driven Parsers

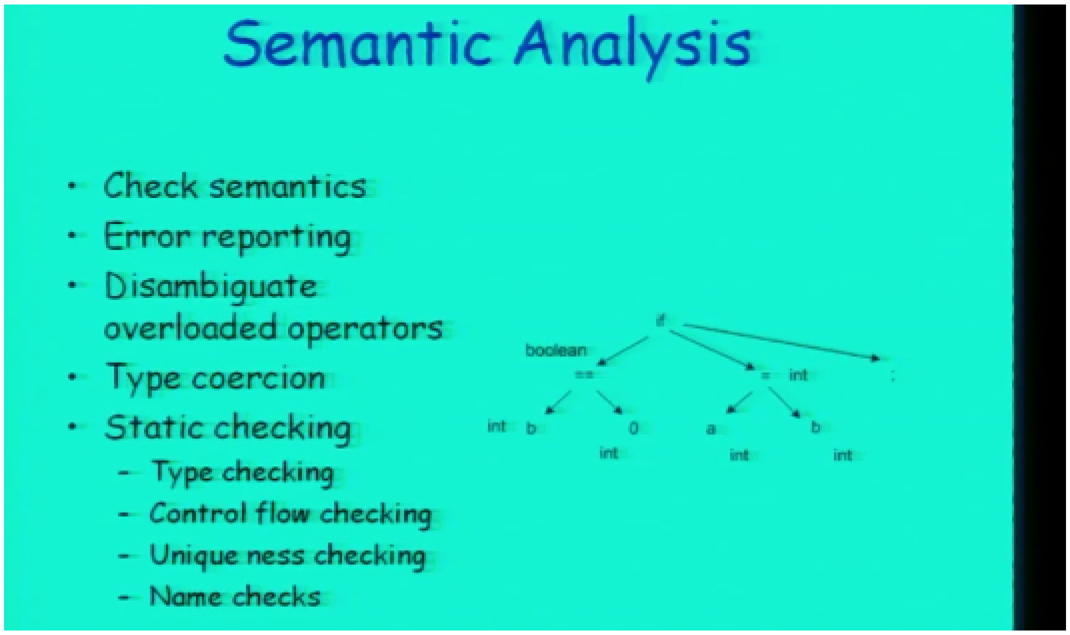
What recovery means is that not only I will flag this error but I will skip this and will try to analyze best of the program to find out more as, okay, so we look at error recovery in more detail later but we actually do syntax analysis, okay but recovery means that I want to read as much program as possible to find more and more errors in the program, okay, and how do I model it? I'm going to use context free grammars to model my syntax analyzer and you already know all the theory of context-free grammars, and how do I implement this? I'm going to use push down automata or table-driven parsers to model this, okay. So this is where you will find that whatever you learnt in your theory of computation will come into play and what happens in semantic analysis phase, this is where the context or information will come in, so I'll start checking now meaning of the program, I'll obviously report errors so if I have situations like this where we are trying to write expressions and trying to do a mix of different types which are not permitted by the programming languages, I’m going to report that and also disassembly going to disambiguate overloaded operators, okay.



Semantic Analysis

- Check semantics
- Error reporting
- Disambiguate overloaded operators

So for example when I write something like this, what is the meaning of this plus? What is this plus? Adding what? So if I write something like this B + C or A+ B and these are two floating point numbers, then this is what is it adding? It's adding two floating point numbers, but suppose these are integers then what is it doing? Okay, now what about a situation when I say I have two strings S1 and S2, so there is you can see that in three different contexts the same one sign, plus sign has few different meanings, okay, and the machine operators are going to be different for this and therefore I will have to disambiguate that whether this is concatenation, whether this is the floating point addition or this is integer addition, okay. So these are overloaded operators and I want to disambiguate, I want to understand what is the exact meaning because at some point of time I'll have to go and map it on to the opcodes, which are availability, okay, I'll also have to do what we know as type version, so if I take a situation like this when I say that I am trying to add an integer and a floating point number and you know that representation of integer and floating point numbers on the machine are different, okay, so I'll have to change this representation into floating point and then do the addition, right. So I have to force this side from integer to float, before I can do really this addition, so that also is a job which is going to be done by the semantic analysis phase, and I'll have to move type checking, I'll have to do control flow checking to make sure that I don't jump into middle of the controls, okay. I'll have to, depending on the language specification I may have to say that no variable can, and one context can be declared more than once, each variable is a unique, variable in each scope, okay. I may have to do name checks, so if I have labels for block, like data says that each block must have a begin label and end label, I want to make sure that those labels are same and so.



Semantic Analysis

- Check semantics
- Error reporting
- Disambiguate overloaded operators
- Type coercion
- Static checking
  - Type checking
  - Control flow checking
  - Unique ness checking
  - Name checks

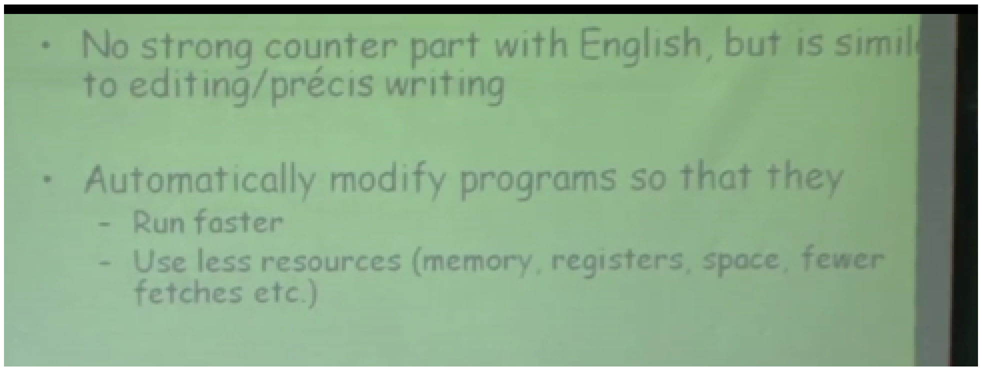
if
boolean
==
=
int
;
int
b
0
int
a
int
b
int

So this is another thing we'll have to do, and what will I get as an output? I will get disambiguated parse tree or abstract syntax tree, so here I will say that each of the operators I am using it will have a unique meaning, so it will not say oh it could be concatenation, it could be your addition of floating point addition, but it will say that I have an integer assignment here, A and B are integers and then I have a boolean here and so on.

So what you get at the end of semantic analysis is? A completely disambiguated abstract syntax tree, where everyone will be able to come up with only one meaning and that is not going to be ambiguity above it, and this is what we know as front-end of compiling, and out of front end of compiling.

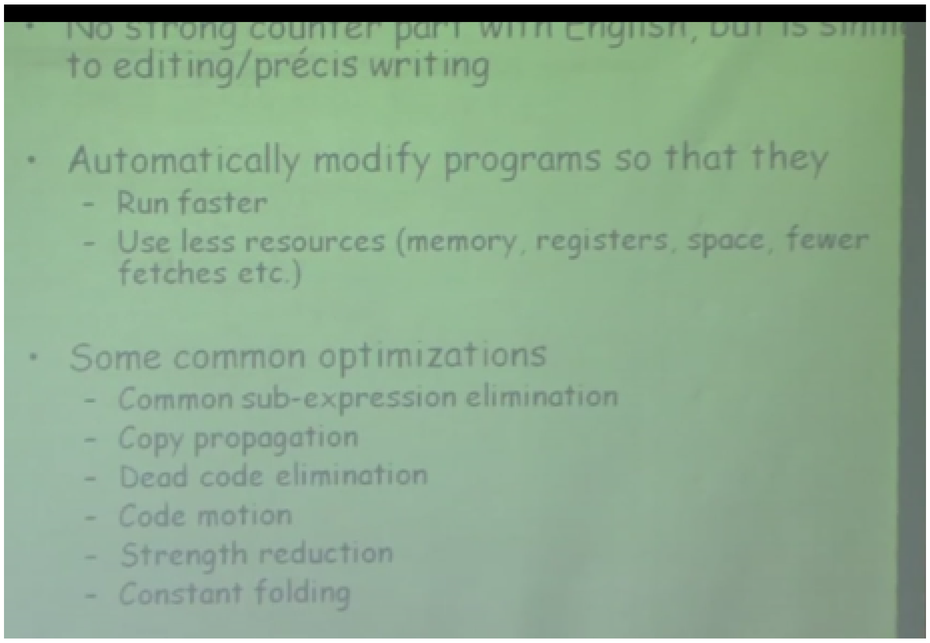
So what does that mean? Give me a concrete example of what you have in mind? Yes, okay, so question is can I have different parse trees for the same input? Yes, we can, right and therefore we have to make sure that when we write grammar, so remember that when I talk of these

languages and when I talk with the grammar I can write multiple grammars of the same specifications and therefore I have to make sure that my grammar is written in a manner that I will not get more than one parser, okay, and that will take care of all the ambiguities, right. So is question-and-answer clear to everyone? So question was can I have specifications it will give me more than parse tree, and answer is yes you can have specification but that's not a good specification and what we'll try to do is either we come up with rules to disambiguated or write specifications in a manner that I'll not get more than one parse, that is what we need to do in the syntax analysis, okay.
So let's move on now and what we have done here, so let me just replace this and say that what I have done is now, I have all this front end pages, okay, but what is it that I do beyond front-end, because I have still not reached close to where I wanted to reach, it was the machine specification or something which is close to machine, I'm still dealing with the source level specifications which are programming language specification, okay. Now we also want to do port optimization, now port optimization if I because I am dealing with, if I mean at least I was giving you similarities with the English language, I cannot find strong similarity except that this is like doing a technical editing or doing a persuade writing when we said remember persuade writing in your school time, you start with a paragraph and say reduce it to almost like thumb rule one to reduce it to almost one-third keeping the meaning same and using fewer words, can I do that, okay. So that is the closest similarity we can find out with what we do in optimization and the purpose of optimization is that we want to have a program which will run faster and we also want to have a specification which will say that it will use fewer resources of the machine, okay.



- No strong counter part with English, but is simil
  to editing/précis writing
- Automatically modify programs so that they
  - Run faster
  - Use less resources (memory, registers, space, fewer fetches etc.)

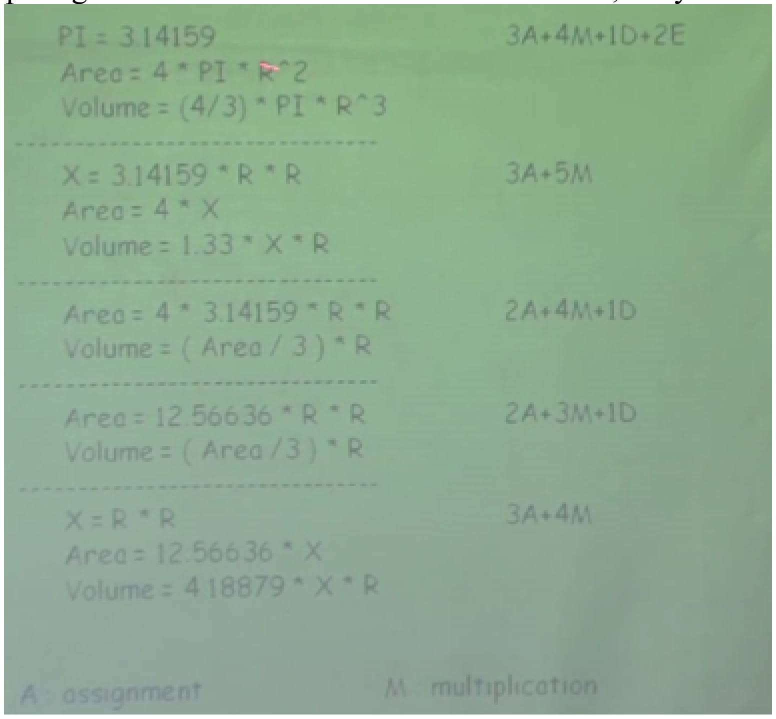
Now you talk of resources which are available to us, memory, registers all these are going to be resources for us, okay, so we want to make sure that I use fewer resources and I have a program which runs fast, okay, but remember that when I say it runs faster, it is again not with respect to the algorithm, it is only with respect to the representation and I'll give examples of that, okay



- No strong counter part with English, but is simil to editing/précis writing
- Automatically modify programs so that they
  - Run faster
  - Use less resources (memory, registers, space, fewer fetches etc.)
- Some common optimizations
  - Common sub-expression elimination
  - Copy propagation
  - Dead code elimination
  - Code motion
  - Strength reduction
  - Constant folding

and there are some very common optimizations which we do at this point of time I'm just mentioning them and very quickly I am not going to give you details of these optimizations but what this says is that when you say common sub-expression elimination this saying that if there is a certain expression which has been evaluated once you don't want to evaluate it over and over again, so for example if I say A is assign B + C and then I say X is assign Y + B + C then I should be able to figure out that I have already computed B + C and there is no need to compute B + C here, provided conditions are met which means that after this computation value of B and C have not changed, okay, if they have not, then I can use the previously computed value here I don't have to recomputed it, okay I want to make sure that something like copy propagation happens that if I am just keeping copy of the variables I may as well use the original variable, okay. I want to eliminate all the dead code, any code which is not reachable during the execution flow of the program that should not be kept in the output and that should just be eliminated. I give you an example where if a condition you know at compile time really well, it's the true or false, and then you know for sure that one part of your ex-statement is not going to be executed you can just throw it out, okay.
Code motion says that if I have a loop and there is a statement which is constant, which is invariant of the loop which does not change its value with a loop execution you may as well move it out of the loop, because it is going to execute multiple times as part of the loop, once I move it out it will execute only once and then I have strength reduction which says that instead of a costlier operator, computationally costlier operator use a cheaper operator and constant folding says that if there is a constant expression which you know at compile time, you may as well compute it rather than here inputs, so if somebody writes code like this, I don't want to generate an add instruction because that is going to again take time I may as well replace this by at compile time itself, that's going to save me one computation. An interesting part is that all these computations actually give you a lot of speed up, these small things which happen in the program they can give you a lot of improvement here, but this part will delay, okay only towards then we talk about optimization but what we want to do is now, okay so here are some examples of optimization, interesting, okay, so these are some common computations if you recall, okay, you have some value which is being assigned to PI and then your computing area

and your computing volume, and if you see how many computations I am doing, I have 3 additions, 4 multiplications, 1 divisions, and 2 exponentiations right, the cube and the square, and you can find that I can do some very trivial things here, you can see that if I look at 4 PI R Square, 4 PI R Square is the common sub-expression which is already computed it, okay. So if I can replace this by something like this saying that I already know that the sum, so I can actually give you many, many versions of this, okay and let me give you all possible versions and you will find that each version has its own advantages and disadvantages, okay, so you can see that basically what it says is that if I just propagate this constant value then I know that we instead of every time computing 4 PI I can instead use a variable like this, okay.

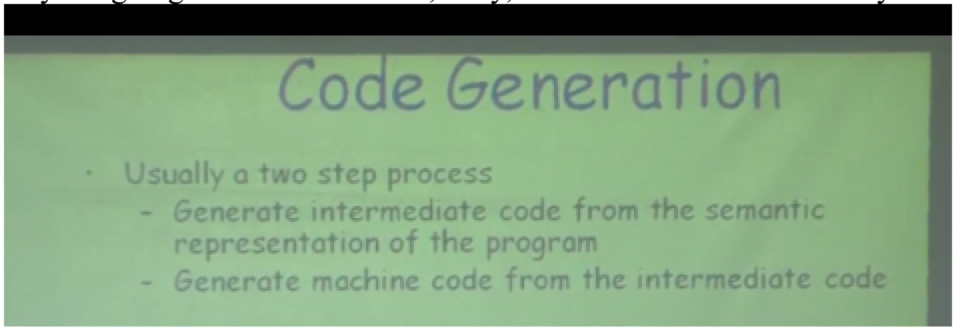


```
PI = 3.14159                          3A+4M+1D+2E
Area = 4 * PI * R^2
Volume = (4/3) * PI * R^3
-------------------------------
X = 3.14159 * R * R                   3A+5M
Area = 4 * X
Volume = 1.33 * X * R
-------------------------------
Area = 4 * 3.14159 * R * R            2A+4M+1D
Volume = ( Area / 3 ) * R
-------------------------------
Area = 12.56636 * R * R               2A+3M+1D
Volume = ( Area /3 ) * R
-------------------------------
X = R * R                             3A+4M
Area = 12.56636 * X
Volume = 4.18879 * X * R

A  assignment                 M  multiplication
```

Now I would also make sure that all my programs are written this way, because this is highly readable remember that when we talked about writing programs, these programs are written for the users, okay and we want to make sure that these programs are readable, okay, so I don't want to write these kind of programs, because I want to make sure of readability of program but I want to make sure that compiler all the time instead of doing so many computations will do something so that I do fewer computations, okay, now which one is the best, okay, that will depend upon in which order compiler is going to do the optimization, but you can come up, you can see that this representation can be change into many in representations but we can also see that as far as this method of computation is concerned I have not change anything, okay I'm only doing pure computations but I'm still using the same formula. So this way compiler does optimization by saying that I can do constant propagation, I can do a constant folding and so on, okay, common sub-expression itself, okay.
So let's move on to the code generation, so this is where somewhere I will say that optimization will fit here, okay but again you can see that optimization has not changed anything in my representation, okay, whatever my representation was before optimization that representation still remains the same only thing that is happening is that I am doing fewer computations to achieve this, okay, but what about this part, because this is where I am going to now change my representation to machine representation, and that is also known as the code generation phase,

okay and error as I was talking in the previous class he said that they are going to be different levels of abstraction in the program so when I talked about source abstraction I was dealing with variables and then I was dealing with operators with this I was able to constitute expressions and once I had these expressions then I went on and I had conditionals, I had iterations using which I made functions, okay. And depending on the language that you can make losses also, okay, so this is really the abstraction I use at the level of source, but what is my abstraction of the level of target? Here I was using memory locations or I was dealing with registers or I am dealing with stats, okay these are typically resources you will have on the machine side then you will have opcodes and then you will have, after opcodes what do you have? You have addressing modes to access all these locations, so what I really want to do is I want to do this translation, I want to change this representation to this representation, okay. Now you can almost see that if I'm careful I have almost a one-to-one mapping here, okay so for example if I say variables, variables are going to be capped either in memory location or register or they are going to be in some stats, okay, so what we do is we now say that again we

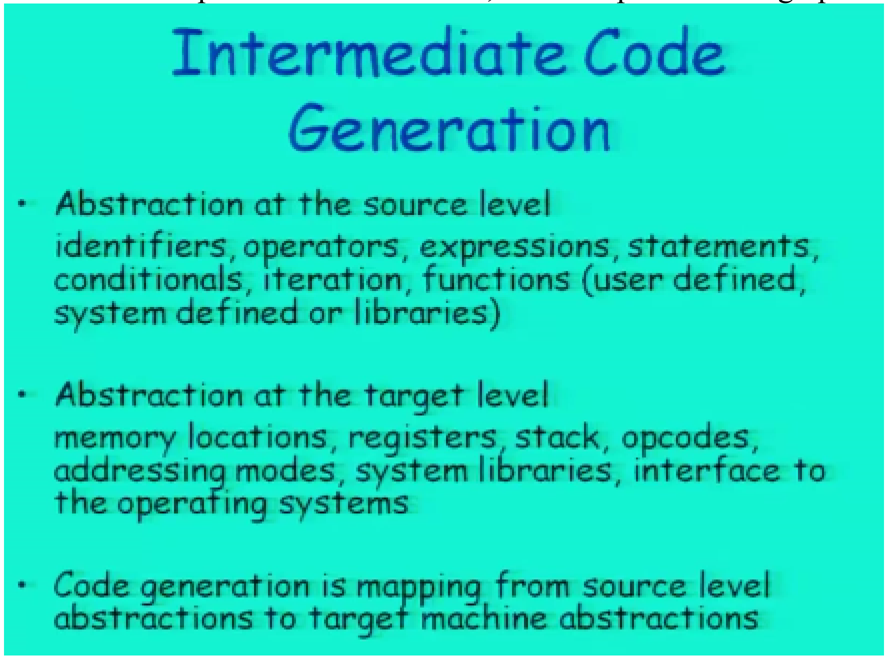


Code Generation

- Usually a two step process
  - Generate intermediate code from the semantic representation of the program
  - Generate machine code from the intermediate code

keep a two-step process because it is possible that when I am dealing with the machine okay, I may be dealing with a class of machines which may look very similar, okay so I may again as I said that may be too tough to jump from one representation to another, in a single step and I want to use multiple steps here and additionally what we have used is an intermediate representation, and this representation is something which is going to be closer to machine but not exactly machine, it will not be complete syntax of the machine, so we want to generate machine code from the intermediate representation and we want to generate intermediate code from the representation I got after the semantic analysis, okay because remember that optimization could be an optional phase, they may be compilers where I may not do any optimization, my program will still run correctly, it may take little more time, it's normally their compilers, this is essential part, this is the essential part and this is really the optional part, depending upon how much effort I am going to or how much effort I am willing to put into building this compile, okay, so advantage is that obviously that each phase is going to be a simpler phase, and it requires therefore that I design an intermediate representation, okay, so my intermediate representation in front end, what representations did I have in the front end? I had a sequence of tokens, I had a syntax tree, and I had disambiguated syntax, okay.
And here is now another representation which will come between the expect, with the dissembling with X index 3 and the machine representation, okay, so there is one more representation is coming up, and most compilers they will perform translation between a sequence of intermediate representation, so generally when I look at intermediate representation or representation which is the source and the final representation which is really the target, they are going to be a sequence between 2 and this is order is decreasing level of abstraction, so this is highest level of abstraction and this is the lowest level of abstraction and everything is coming closer to the machine, that is what we really wanted, okay.

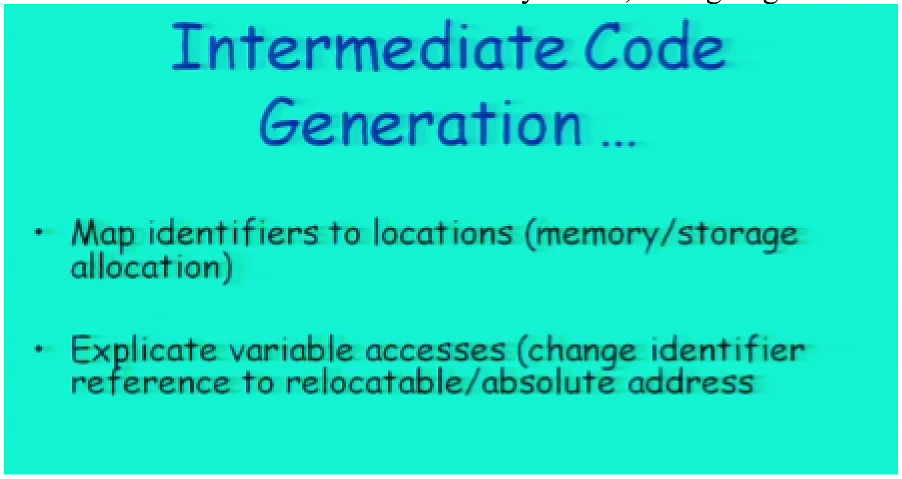
So typically one after another representation code is the most important thing to do here, now what is it that I really want to do? So let me pick up a complete example of intermediate code okay, so my abstraction as I just decide there are identifiers, operators, and so on and target level I have all these memory locations and so on, so first thing I need to do is I need to take identifiers then put them in either memory or in register or on stats, okay so this is what I call as memory allocation or allocation of resources, and I'll say that this variable is going to reside in such as such memory location, this variable is going to register and so on, that's the first thing I need to do.
Now once I have identified, I have put locations for each identifier then I want to see that corresponding to this operator what is the opcode I have, okay. Now on many machines you will find that almost for most operators we have today in the programming languages you will be able to find an equivalent opcode, you'll be able to find that this particular operator can be mapped onto this opcode, but there could be situations where you may not be able to find an equivalent opcode. So for example if you say I want to do string concatenation, it's possible that on your target machine you may not have something similar so what will you do in that case what will I do if I want to implement concatenation, for example or a string operation,


Intermediate Code Generation

- Abstraction at the source level
  identifiers, operators, expressions, statements, conditionals, iteration, functions (user defined, system defined or libraries)
- Abstraction at the target level
  memory locations, registers, stack, opcodes, addressing modes, system libraries, interface to the operating systems
- *Code generation is mapping from source level abstractions to target machine abstractions*

I just need to write a metro, right so if you know your assembly language you just write an equivalent metro, which will do the same thing. So at some point of time they would microprocessors which did not have multiplication or division, so how did we implement multiplication and division on that? By a sequence of additions or subtractions, okay, so you can always find out that either I can map this particular operator to an op code or I can write a metro to implement this particular op code, okay.
What happens to my conditionals here, so I have these statements and then conditionals and iterations, how do I implement conditionals and iterations, so is there something special in conditions which I need to implement at the machine level, what is the equivalent abstraction here? Branching, but how do I do branching on the machine? Jump, for testing jump, right, so if you have a test in jump I can just find out I can do comparison of two locations so I can say if A = B what that needs to be done is I compare A and B whatever locations are now bound to A and B and then depending upon whether A is less than B or A is equal to B or A is greater than

B some conditional course are going to be set and then I'm going to do a jump, and how do I implement unconditional jump or go to just jump to a level, right. Because every opcode is going to, or every instruction is going to be in certain locations, so I know the addresses and I just have these jumps, whenever you have these functions either these are user defined functions and therefore they can be just converted into transfer of parameters and then jump to a location or I can do system libraries, right, so I mean using system libraries here or mathematical libraries in so, or I may be using OS, so depending upon what kind of calls you have here, you will have to find an equivalent all at the machine end, okay. And this is what the code generation is. So code generation is really mapping from source level abstraction to the target level abstraction, and the first step here was that I am going to map all the identifiers, so I am going to take these identifiers and allocate memory to this, I am going to also now find out.

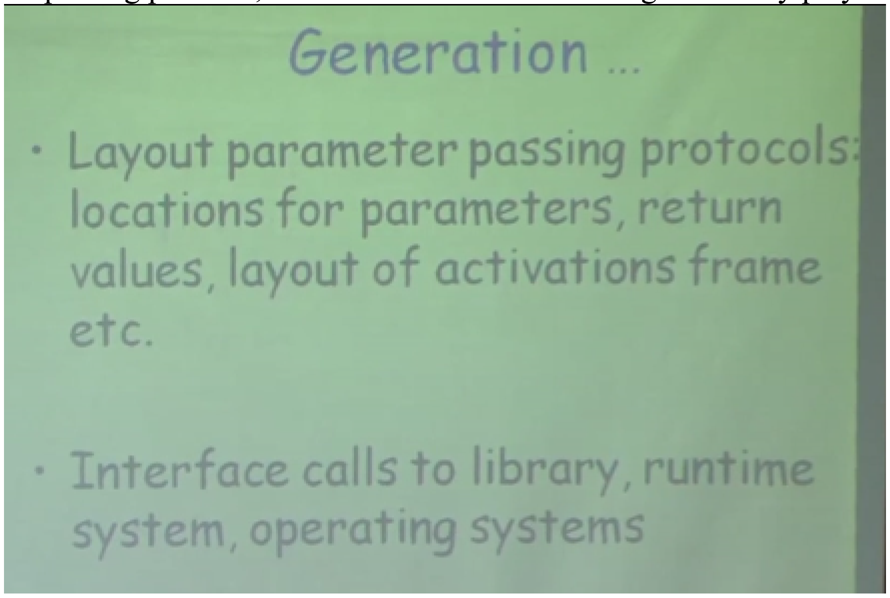


Intermediate Code Generation ...

- Map identifiers to locations (memory/storage allocation)
- Explicate variable accesses (change identifier reference to relocatable/absolute address

So this is an important point to understand because I did not talk about really the addressing mode, okay, I want to explicate variable accesses, so what that means is that when I say an identifier, identifier is never going to be in a fixed location, okay. Now depending upon the scope for example suppose this is I am trying to say that A is going to reside in certain memory and A is part of, is variable which is declared in the function. Now at some point of time function will be taught, now can I find a fixed memory location to A, that would be very inefficient, because only when function is called I don't know how many other functions are active that point of time I want to allocate some space to it, okay, but I also want to know that A is going to be fixed at a fixed location with respect to certain offset, so I will always say that if A is part of a function, then doesn't matter where this function is allocated space whatever is the base address of this function with respect to this base address A will be an offset of 100, some such number right so I can find out this information that what is the offset of this particular variable and what is going through the base address, and what does explication mean is, that a time of code generation I want to change all these identifier references to either re-locatable addresses or absolute addresses, okay.
So what is going to be re-locatable address of A? Re-locatable of address of A is going to be some offset with respect to a base value, okay so you know that's indexing so for example if this is an array and if I say I want time to access A5 then what will I say? I'll say that there's a base address of A with respect to certain other base value which is coming from the context information and the index value is 5, so I am going to use a complex addressing mode to find out what is the location where I can find out value of A5, okay, so I use base address of the total scope within that base address of A and within that and index of 5, right. So this is how I start using all my addressing, okay. So I want to map all the operators in opcode 2, a sequence

opcodes and I want to convert conditionals and iterations into a sequence of test index, okay. So intermediate code generation is that I want to also, at the same time lay out all this information about parameter passing protocol, because function is something that really plays a very


Generation ...

- Layout parameter passing protocols: locations for parameters, return values, layout of activations frame etc.
- Interface calls to library, runtime system, operating systems

important role, okay and we will see that how do we do this part and then we also have ways to handle all this return values are going to be handled then or we lay out all these activation things.

Right now these may all sound like terms, which don't make much sense but when we talk about intermediate code generation, this is going to become very important, okay and that point of time it will become more clear, and then we also want to have interface calls to all the libraries in runtime system and so, okay. So let’s break here today, and we’ll continue our discussion from this point onwards in the next class.

Acknowledgement

Ministry of Human Resource & Development

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

Shubham Rawat

Shikha Gupta

Pradeep Kumar

K.K Mishra
Jai Singh
Sweety Kanaujia
Aradhana Singh
Sweta
Preeti Sachan
Ashutosh Gairola
Dilip Katiyar
Ashutosh Kumar

<u>Light & Sound</u>

Sharwan
Hari Ram

<u>Production Crew</u>

Bhadra Rao
Puneet Kumar Bajpai
Priyanka Singh

<u>Office</u>

Lalty Dutta
Ajay Kanaujia
Shivendra Kumar Tiwari
Saurabh Shukla

<u>Direction</u>

Sanjay Pal

<u>Production Manager</u>

Bharat Lal
an IIT Kanpur Production