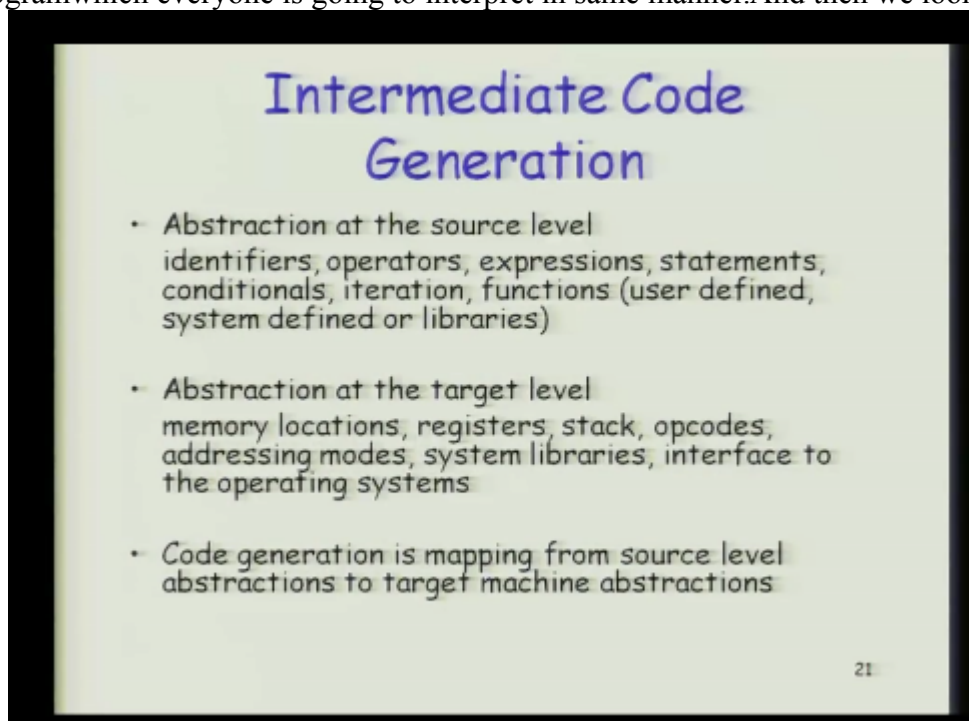


Indian Institute of Technology Kanpur
NP-TEL
National Programme
On
Technology Enhanced Learning
Course Title
Compiler Design
Lecture – 03
By
Prof. S.K Aggarwal.
Dept. of Computer Science and Engineering

What are the various phases of the compiler? When we look at frontend phases, and we looked at like semantic analyzer and semantic analyzer and what are we doing by end of this process was, capturing the meaning of the program and we were able to get a unique meaning of the program which everyone is going to interpret in same manner. And then we looked at



certain optimizations and once we get optimization we probably moving to different things from the program without running about what the underline generation was, then we started looking at the completions of the abstract syntax tree which was disambiguated already by the semantic analyzer phase, it was something which is closer to machine and we looked at abstractions which are available to us at the source level. What's this abstraction at the target level? Then we were looking at how to convert now these abstractions into the machine abstractions, right.

And you said that process is going to be that all the identifiers, able to meet into certain locations and then all the variable accesses depending upon where these identifiers were, you were going to find out exact locations from where we could pick up all this information

Intermediate Code Generation ...

- Map identifiers to locations (memory/storage allocation)
- Explicate variable accesses (change identifier reference to relocatable/absolute address)
- Map source operators to opcodes or a sequence of opcodes

basically all the addressing mode information will come here, and then we want to map all the source operators to the target operators and if there is no equivalent target operator then we want to write some kind of matter for that, and then we want to convert all the conditionals and iterations into a sequence of testing jobs, okay.

And after that we started looking at the parameter parsing protocols and this is where, please sound especially is required, so what we want to do with parameter parsing is that we want to find out where are we going to put all the arguments and we want to find out where we are going to pick up the return value from, okay. So typically what is going to happen is that if I say call some function P, make a list of actual arguments, what is going to happen that some of them hit my machine, I'll have to say, what is the place where the arguments are going to be, and I also want to find out the place where I am going to put the return value, okay. Now I cannot put all this information at some argument place, I need to put it at some fixed place and therefore we must have certain protocol which will say that there is a parameter passing, how I am going to look at passing, where I am going to put all these

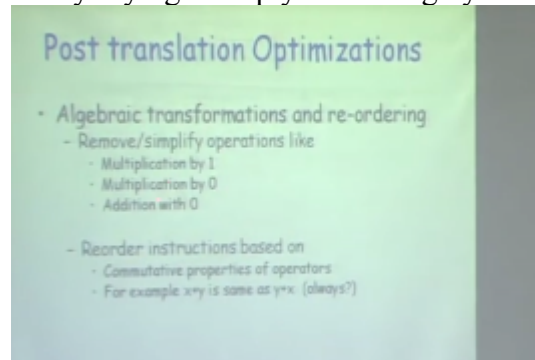
Intermediate Code Generation ...

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

information, what is going to be the layout of this procedure, where I am going to put activation frame and so on, okay.

so I must layout the complete activation frame of the functions, and when we do runtime system and we talk about code generation and talk about the code generation for procedures and functions that time the layout of activation frame will become clarity, okay, and then you also

must have some kind of interface to all the libraries which may have, so you may have mathematical libraries, you may have statistical libraries and so on, you may have call to runtime system, operating system and so on, so we must know how to make the system, so compiler will have to find out appropriate that input, right, everyone is instinct at this, okay. So once we do this, okay, then like we had this file of structure to begin with, we said we are going to have frontend phases, after frontend phases we had certain optimization then we started talking about code generation phases, okay, we can also do a little bit of optimization after code generation, in some optimization we were trying to do was, we were trying to remove all kind of redundancy which are present in the program, but some redundancies do come because of the code generation phase, when I'm trying to do code generation certain more redundancies will pitting and I would like to remove them and what kind of redundancy maybe, so I may generate information by saying multiply something by 1 or multiply by 0,

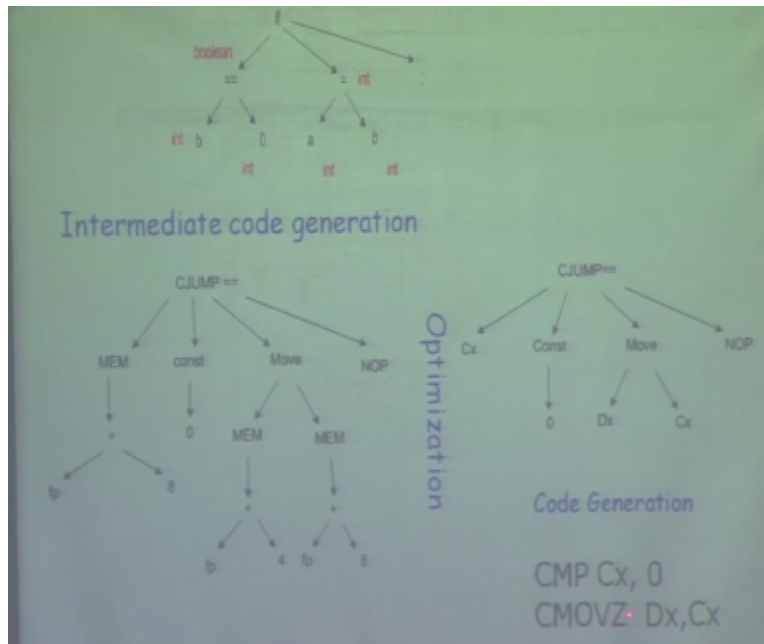


okay and typically this kind of information will get generated when I'm trying to do code generation for either code is or for structures and records, okay, and that point of time if some such information gets generated, we would like to just remove them by doing a post code generation optimization.

And we'll also try to use commutative properties of operators we have, so for example if I have an expression like this, what's this? An expression like this, when I say $X = X + Y$ or $X = Y + X$, okay. Now you will see I try to use any C compiler and generate code for this by switching off all the optimizations and you will find that the two code sequences are going to be different, and what could be reason for that? By these code sequences may be different, so normally my compiler is going to send input from left to right, okay and then it will be able to figure out that the first argument is same as the left-hand side and therefore once I evaluate the second argument or the remaining arguments I'll just pick up that value and add it to location where X was already stored, okay but when I am sending this it will first encounter Y, then it'll encounter X, then now X may be somewhere in the middle, anywhere, okay, so I put even of this form, and therefore it'll start loading this, start loading this, will add the two so on, okay, so normally what we are trying to do is this is the best possible way and in fact C already uses a very concise notation for this and they have introduced an operator line, right. So the whole idea of writing a sequence like this is that I can generate better code.

So now I am going to actually what I am doing here is I am using certain properties of algebra here, saying that $X + Y$ is same as $Y + X$, in most cases unless I am going, I proceed the medical analysis, okay so this is really that's why I put this question mark, if sometimes when you are trying to hide a perceive a numerical analysis then if you try to use the protective properties it's possible that you'll end up getting different results and such optimization, and all this, okay. And then I'll start actually using construction selection so once I have done the intermediate code generations then I'll look at the complete text of the machine and we will try

to find out how do I use the syntax of the machine or finding out addressing mode, opcode, and the peephole optimization and so on, okay. And once that is done, okay, what will happen, so this is where we were at the end of the frontend, okay, so I have the same example running through, this is the disambiguated aspects syntax tree, we had at the end of the frontend and now what happens is when I try to do intermediate code generation, it generates a code like this which is not with

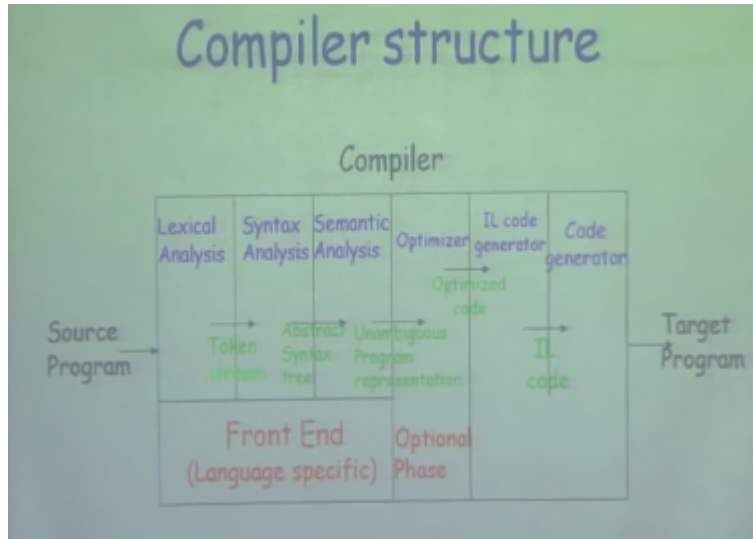


respect to any machine which is not by particular machine, but it's close to many machines, so I may have something like this which may say that this variable P is actually stored at an offset of 8 with respect to certain frame pointer, and this is constant 0 and when I add 8 with the value in the frame pointer, what does it give me? It gives me address off where we stored, but I need the value of P, so I do a P reference here, I do a memory P reference and that gives me the value, okay, I add the two and similarly on the right hand side when I look at A is different offset of 4 from some frame pointer, B is code again at the frame pointer with the offset of 8, I add the addresses B reference and then add that, okay. So this is assignment, okay so this is where move takes place, so I'm just taking to slightly moving in to this address and then I'm using an X statement here which is equivalent to conditional jump and so on, okay.

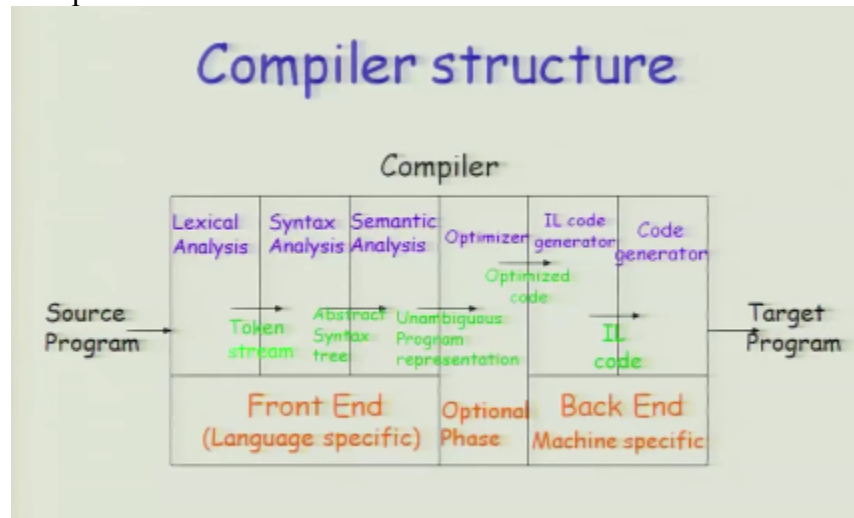
So there is a kind of intermediate PI we get, and when I do an optimization, now optimizer they say that I am going to put all these values in a register and not into memory location, hence so I may have registers like saying that this will occur in, B will be in register CX, and A will be in the register DX, okay, and therefore some optimization happens here, I don't have to do all of these, memory dereferencing. And then finally I may end up generating code like this, and this is compares CX with 0, and move on 0 CX to DX, if it is 0, then just move it, okay. So starting from whatever I started with from expression, I may finally generate code like this after going through many transformations from the book.

Now you can see that all I have done is that I have change the representation from whatever I started with, but I am actually employing the same measure and I'm not changing anything, okay, so how does my compiler look now? This is where we left our compiler with the frontend and we left this part of the block empty and if I now try to fill it in we'll have an optimizer which is an optional phase and then we'll have two phases of code generation, one is intermediate

language code generation and then we have code generation and this also is a historically or traditionally



known as the backend trees which is specific to the machine, so this is where the machine specific information starts coming into the compiler and this really completes your overall structure of the compiler.



Questions/comments, anything? So I can avoid, we have no clue of what happens really here or how lexical analyzer works, how syntax analyzer works and so on, okay, but we've some idea of that if I take a representation here and I apply these phases then what would be the representations here and what would be my final representation? Okay, at least that part is sort of clear to everyone.

So let's move on, okay and now what happened now is that there is

- Information required about the program variables during compilation:
 - Class of variable: keyword, identifier etc.
 - Type of variable: integer, float, array, function etc.
 - Amount of storage required
 - Address in the memory
 - Scope information

some information which is lacking here, okay, so for example when I look that something like this, okay so let's just take a piece of code, so if I say $A = B + C$, okay, this is probably which I'm trying to get it code, when I start converting this into a lexical analyzer, okay what will be the sequence of constitution generated by the lexical analyzer, which will effect to the syntax analyzer? This is the input, right? So what is the sequence, so if this is my input, this is the sequence of input I have, which I have send to my compiler, what will be the outcome of lexical analyzer? So what is it, then I'll see here what will be my token here? Doesn't matter, what is the types are, right, do I need to know types and find out what my tokens are? I don't need to know the types. We need to define, that will be declared offset, somewhere of to the token, right. So I'm only looking at this part of the program and say tokenize this, what will I get this total? One by one, somebody can raise the hand and then I can, so 50 of you start speaking I won't be able to figure out, yeah.

Whereas A operator equal B operator plus C, is that what it will be? Add some white spaces, but white spaces do not want anything to the mean, right, so I can ignore white spaces, I don't have to worry about the white spaces, okay, really this is not what happens. Now what happens is, I didn't say that either now identifies, now as far as structure is concerned, as far as syntax analyzer is concerned to check structure whether this is a valid expression or not, whether this is a valid assignment or not, it need not know anything about what the variables are, right, all it needs to know is whether this is an identifier or not, so what I don't like to pass on, I'd say identifier assign, identifier some plus of operator which is addition operator and again an identifier, okay.

Now as far as my syntax analyzer is concerned this information is sufficient to do structural analysis, okay. When it comes to type checking at that point of time it will need to know what is the type which is associated with this identifier, what is the type which is associated with this, what is the type with this, and what this exact operators, this is what my semantic analyzer mean, when I come to code generation when I say now assign certain memory locations to this that time it will say oh this particular variable is put into certain memory location so I will say that this one in an offset of A, this one is offset of, so let's say this was 4, this was 8, and this was 12, okay. So this set lot of information, if I look at this identifier, there's lot of information which is associated with this identifier, one information is really the string, which is this thing, okay, which I need to know that the string associated this identifier different than this, okay. I also need to know the type, I also need to know the offset and maybe some other information like the type I will also get by default information saying how many bytes it is going to take when I store it in memory, right. Now where do I store all this information? If this is the sequence of tokens I am passing, I can't effort to lose this information I leave it at time of code generation, where do I store this information? I need to store it somewhere, okay, and for that what we have is that information which is required about the program variables during compilation so I may have various kind of information with rather

- Information required about the program variables during compilation
 - Class of variable: keyword, identifier etc.
 - Type of variable: integer, float, array, function etc.
 - Amount of storage required
 - Address in the memory
 - Scope information
- Location to store this information
 - Attributes with the variable (has obvious problems)
 - At a central repository and every phase refers to the repository whenever information is required

this is the keyword, this is the identifier, what is the class, what is the type, and all the storage requires, address in the memory and so on. This information is either it can be right here so I can define this as a structure and I say that all this information write it, but then you can see I get into obvious problems and what are obvious problems? That's suppose I change this to A what will happen, this information we get duplicated, if I change anything I need to go and find out all these places where this information was and change that, okay, or I can say that I have something called a simple table and I can say all the information corresponding to A is here and then all these are pointing to A here, okay. So if I now create another data structure we have information about all these are going to be stored this is where we store it and this is the data call simple table at this point of time, we will not worry about what actual data structure I employ, you can just assume this is an idea of records to begin with, okay, where each record is going to have information about each of these symbols are have mapped over, okay, so how do you my compiler look now. That along with these phases of compiler, I must have a simple table, where are the simple table can interact with all the phases, okay. So what each phase will do, lexical analyzer will not have type information, for example it will just say, what is the lexicon which is associated with the variable, so only thing that's lexical analyzer will know is this identifier has a lexicon associated with this which is a 8, right.

When it comes to type checker, type checker will figure out what is the type of it and will put this information when it comes to code generation of memory location, memory locator will say if type of this is floating-point I need to allocate it bytes for this, and memory allocator will immediately say that since I have possess let me assign certain address for this and it will give me the phase address, when it comes to code generation, code generator will again figure out where I need to put this information and I may need to associate certain registers with this, okay, so all that information will keep going in the simple table and therefore you see bi-directional arrows here that all these phases will be able to write into the simple table and will be able to read from the simple table.

So this is additional information I need, in addition to the flow of data in my compiler, yes, why? Oh so if I start reading this from left to right if I encounter A first, and lexical analyzer figures out that this is an identifier, okay, now where do I store string 8, this S8 has to be stored

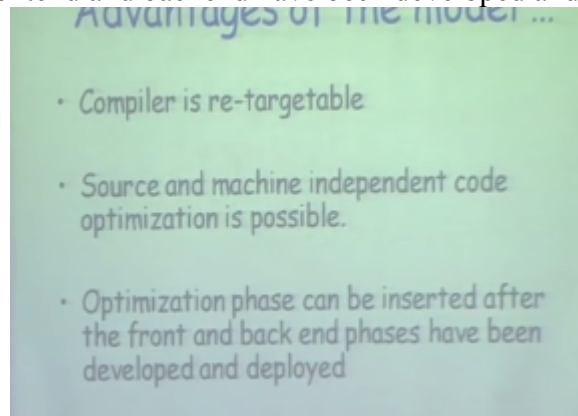
somewhere, where do I store it? In the simple table, so if I cannot write into simple table how will I store it? Okay, similarly when I come to this phase which is semantic analyzer phase and this figure out that type of A is B here, where do I store this information? I need to write it in simple table, okay, similarly when it comes to memory locator, memory locator says that I'm going to assign this variable at offset of 16 from some frame pointer, where do I store this information? I need to go back to my simple table, okay, so I need to be able to write, I should be able to read from the simple table, okay. So not all the information will be written by all the phases, each phase is going to write some information which is relevant to that particular phase.

Is your question answered? Okay, so here is a model of compilation here, okay and interestingly as I pointed out when we were talking about history of compilers, that this was the model which was used exactly as in 1957 compiler which was first Fortran compiler, okay, that structure has not changed, last 55 years, okay. Now what are my advantages? There must be certain advantages otherwise you would have changed all the structure by now, right, so what are the advantages of this structure and what are the disadvantages of this structure? Code of compiler or code complete compiler, so code will be modular, good, okay, so basically what advantage we clearly see here is that each phase is doing something which is the logical activity which is complete in itself, okay, so like lexical analyzer is just tokenizing the input, right, my syntax analyzer is just taking structure, my type checker is just checking the types and nothing else, okay. So you can see that you have high modular 4 where each phase is doing something which is a coherent logical activity, that's really the good part of this.

What are the other advantages you can think of? Other than that, yeah I mean modular we have already listed, so other advantages, different machines or different languages, if you elaborate on that? So if I have one language and I have multiple machines then the same frontend can be used very good, and we should be able to apply to the backend if I have many languages and one machine that I should be able to use the same backend, right. Signs like very nice property that we have been able to employ, so these are the advantages also this is known as analysis synthesis modular compilation where frontend phases are known really the analysis phase, what frontend is doing is basically taking an input analyzing it to find out whether this is a valid input with respect to the language definition or not, and what is my backend will? It's synthesizing the program, which is a machine program for some correcting, okay and each phase has a well-defined work and each phase also handles a logical activity in the process of compilation, okay. Now you can see that I can just take this phase do something with it and in the process of debugging if I find that certain information is not correct, then very precisely I know where to go backend start looking for possible bugs, I don't have to really look at the whole compiler, I just need to focus on part of it, okay, continuing on this okay and this is really what some of you already pointed out that I can use part of the front end, part of the backend and the whole compiler becomes retargeted, so suppose I'm trying to write a C compiler for a new machine I don't have to rewrite the frontend I will pick up frontend as it is and change that, and my compiler gets retargeted to a new machine, okay.

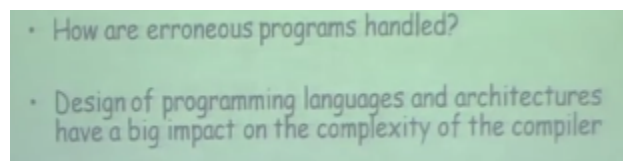
Similarly if I have compiler for a machine and I say a new language has been designed then I just need to write the content for this and then I can work with the backend which is already available, so source and machine independent code optimization is also possible because if you see optimization what I had okay that was independent of both source specification and the target machine that was working on certain intermediate representation. And therefore another advantage is that I don't have to plug in an optimizer to begin with, I can have a full compiler

a working compiler and then at some point of time I can say here is now an optimizer which I am going to plug in and suddenly your code quality improves, same compiler will continue to do work, that's another advantage, right, so highly modular structure and optimization phase can be inserted after the both frontend and backend have been developed and this is actually a clean



model and this is how people do it in industry that you first have a functional compiler which will assure a functional correctness and then go for an optimizer which you are going to plug in at some later point of time, okay.

Now what are the various issues, which you need to be alerted about when we start design the compiler, so what are the various issues we need to face here, structure will fail this stage forward so far, okay, but as we start going deeper into it then you will start noticing the kind of problems we face, now when I take each phase specifically with discuss the problems here of each phase, so I talked about what are the pitfalls you may notice in the lexical analyzer, and what all the problems you may notice in the syntax analyzer and so on, okay but we need to worry about for example suppose your input is incorrect, okay, how do you handle incorrect in it? Now when I say input is incorrect what does that mean, it's not that you have a buggy program, the program could be buggy, could have logical errors but what I'm worried about is that with respect to the language specifications where they look input is correct or not, because compiler has no way of figuring out, whether they are logical errors or not, right. So now you have to worry about incorrect programs you need to do what we know as error reporting an error recovery, okay.



Now as your languages become more and more complex and as your architectures become more and more complex your compiler it's going to have an effect on the compiler I'll not say whether it becomes simpler or more complex, so for example if I want to write compiler for programming language C, what does a compiler for a programming language like C++, the two are going to be very different, the reason is that if I am trying to design a C++ content then I have to do so many things in my syntax analyzer and type checker, okay, that could be virtually held, okay, and then their languages which are even worse than that so if you try to go and Google for language called Chill or language called ADA then suddenly you will find that they're much worse, okay much worse in the sense let me not to use, what something that sounds very negative, okay, they're more complicated.

And obviously some of you right ADA to handle these as your input into projects, okay, some of you will be, not all of you will get to write the C compiler, some of you who are brave hearts, they'll pick up languages like ADA and Chill right compilers. So design of programming languages and architectures are going to have a good impact on complex of your compilers, and then this immediately brings us to this issue of retargeting compile, okay, so typically what may happen is or what is decided, is that whenever there is a machine and whenever there is a programming language I must have compiler for programming language for that machine, okay. And if you say that there may be 10 to 15 popular architectures and maybe 20 popular languages then I need about 200 compilers, okay, somebody will have to sit and code them, okay. Now this is what is also known as classical $N \times M$ versus $M + N$ problem, okay, so what we are trying to do is, yes I have to find out whether there is a situation so some of you really talked about saying

Issues in Compiler Design

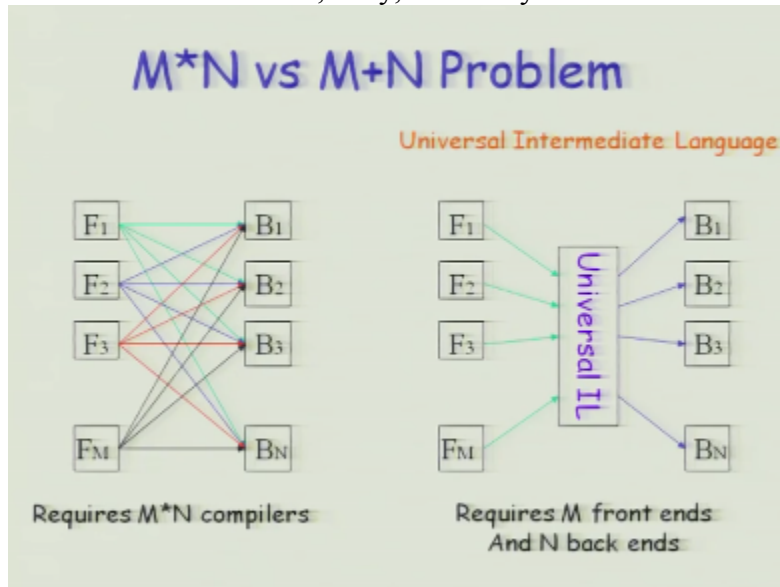
- Compilation appears to be very simple, but there are many pitfalls
- How are erroneous programs handled?
- Design of programming languages and architectures have a big impact on the complexity of the compiler
- $M \times N$ vs. $M + N$ problem
 - Compilers are required for all the languages and all the machines
 - For M languages and N machines we need to develop $M \times N$ compilers
 - However, there is lot of repetition of work because of similar activities in the front ends and back ends
 - Can we design only M front ends and N back ends, and some how link them to get all $M \times N$ compilers?

31

frontend and backend, okay, now if I use that situation what is the most critical part, so here if I say I have these frontends, okay, and I have these effect frontends and I have backends which are let's say going from B_1 to B_N , okay so I have these front ends and I have backends, now to connect to them I need something okay and what is that something I need? I just cannot arbitrarily say that pick up this frontend and pick up this backend and give me a compiler, right there is an intermediate language there, okay so what you're talking about is some intermediate language and this can also be called as a switch box, okay, when you say that all frontends will be able to generate same intermediate language and all backends will be able to work with the same intermediate language of input, right.

So what I need now are just N frontend and M backends right, so instead of having $N \times M$ compilers I can just write, N frontends, and M backends, so $M + N$ and then life is simple, but life is simple so long as I can design or universal intermediate representation, right, so because there's always a trade-off, right, trade-off here is that I should be able to design this switchbox now, or is that straightforward, now suppose you have imperative language then you have logical languages here, logic programming languages you have functional languages all kind of a complex languages may be there and you are saying that I should be able to translate everything into a single intermediate representation, so is that reasonable? Is that realistic? No

then all this structure goes for atoss, right, so what we're talking about I can employa frontend and the backend and so on, okay that will not work, so what do I do? Okay, so typically this is all my N cross and compilers look and this is how my universal intermediate representationwith all these frontends and backendswill look, okay, and reallydo it for this is that I



must have a universal intermediate language, so whatproperty is this universal intermediaterepresentation was at, that it shouldwork for all languages and all possible machines and they doesn't seem to be muchcommonality between this large set oflanguages and large set of machines, sointerestingly this product departed way backbecause people realize this problem wayback when the first compiler waswritten in 1958 itself peoplestarted talking about it, okay

Universal Intermediate Language

- Universal Computer/Compiler Oriented Language (UNCOL)
 - a vast demand for different compilers, as potentially one would require separate compilers for each combination of source language and target architecture. To counteract the anticipated combinatorial explosion, the idea of a linguistic switchbox materialized in 1958

so because atthat point of time itself people were saying that look coming idea is notmachines but other machines to get in developed and at least two, three otherprogramming languages for quickly comingup so COBOL was there people weretalking about ALGOL and some otherlanguages were there, so this idea of a linguistic switchbox actuallymaterialized within a year of the firstcompiler.

An idea was that if I can comeup with some kind of compilers that timeit was called compileroriented language, universal computer language and so on, you can give various names to thisacronyms, computer scientists are verygood in inventing these kind of termsvery quickly,

okay but the whole idea was that I want some kind of linguistics, linguistics switchbox, okay which will be able to then do this translation, okay and supposed you have certain properties and basically idea was to reduce the total development effort of compilers for

potentially one would require separate compilers for each combination of source language and target architecture. To counteract the anticipated combinatorial explosion, the idea of a linguistic switchbox materialized in 1958

- UNCOL (UNiversal COmputer Language) is an intermediate language, which was proposed in 1958 to reduce the developmental effort of compiling many different languages to different architectures

different language to mapping onto different architectures, okay, and they did not succeed, okay. So the whole effort, yeah in every first one proposal was made in 61 to reduce this development effort, yet another was made any how many people were talking about and other efforts were made, and what happened was that, it is next to impossible to design a universal intermediate representation, so if we don't succeed here, do I go back to this model, when I say forget about this box, and have $M \times M$ compilers, no, very good. If no then, what is the solution?

Very good, so really this is, this was the solution which was proposed saying don't look for a general solution, but look for a solution that will work for a subset of languages and for a subset of machines, okay, and this was the idea which was proposed and rather than saying that, so there is many, many illustrated part I have given here which you can read some point of time, okay, the idea was interesting saying don't look for a universal intermediate representation but find out languages which are similar and find out machines which are similar and design now intermediate representation which will work for this set of machines, so I may not have a solution for this $N + M$ problem, but I have this solution for a subset of machines and a subset of languages, and this is what was proposed and so common IR for similar languages and similar machines have been designed so if we look at

- It is next to impossible to design a single intermediate language to accommodate all programming languages
- Mythical universal intermediate language sought since mid 1950s (Aho, Sethi, Ullman)
- However, common IRs for similar languages, and similar machines have been designed, and are used for compiler development

35

GCC for example, GCC will find force on too many machines, it can compile many languages and it has a common intermediate representation, and in industry also when you use many of the compilers they use a common intermediate representation called set of languages, and a set of machines, okay, and that really phase of working solution.

Now other important quick question and important issue that we have to face is, how do we know that the code which has been generated is correct, and we already saw that there is no way I can prove that my compiler is there, okay, now next let's understand the scenario, let's visualize the scenario, okay, when you started working maybe in first year AST101 and which was the first language do you use C or Java C, and when you compile your C program and did not work, what was your first reaction? Check, redo the program, right? Okay, anyone else tried something else? Recompile, but recompile the same program nothing can change, so if the program is not working and you recompile it's the same behavior when that behavior, nothing changes there, okay.

Compile with the different compiler, but when you are sitting on that, when you are sitting on the computer and say what different compiler you have? So you started debugging program, now tell me one thing, I don't want that program error, but you did not have confidence in this program, you are saying my program is incorrect, I need to debug it, did you ever blame the compiler saying my program is correct, I don't know who has written this compiler? It has generated the wrong code and therefore my program is not working, did you ever blame the compiler? That means you have this confidence in compiler, that whatever compiler is doing is correct, what I am doing is perhaps buggy, and therefore I need to check my program, okay, that means somebody must have done or good come of, testing the compiler or good job of convincing you that don't blame the compiler look at your program, so how do you get it this level of confidence in your program, compiler is yet another program right, so can you get that similar level of confidence in your own program and if yes, how we can do it?

I can tell you one thing, now what is structure? When we were doing this course like equivalent of AST101, so I already load the program, our instructor was Professor Salsa Buddha, he will come to the lab and say so are you confident that your program is correct, yes, okay, run it, run it, it will take some input and how do you give input, you just put it hand to the keyboard, okay and most of the time the program is going to crash because some random input will go, so if you are saying really B integer and if your input is not integer what happens, code up, okay you are taking some input but you are not checking whether this input should always be integer what happens if I give character input, straightaway it goes for folder, now that is not how real programs were, if you give incorrect input to compiler have you ever seen message in the compiler, gives you a nice error message, right. So really compiler goes through lot of testing and one way is obviously prove that it is correct, but that is something you cannot do so program proving techniques do not exist at level of where compilers can be shown to be there, okay. So what we need to do is we need to do a very

- Prove that the compiler is correct.
- However, program proving techniques do not exist at a level where large and complex programs like compilers can be proven to be correct
- In practice do a systematic testing to increase confidence level

systematic way of finding out or systematic testing which is going to increase our confidence level, okay and what is normally done is, we have something known as test suite for each programming language.

This test suite is independently designed by language designers, there is not part of the compiler, okay, so what happens is that I have language, I have a language specification, and I have a test suite which is going to test any compiler which is going to be written for this language, okay and what does this test suite contain? This test suite contains thousands of programs, okay sometimes it can run into tens of thousands of program and each program is going to test a specific feature of the language and will have a documented behavior saying if this is my input this, this should be the output of the program and it'll also have buggy programs which you say if, this is my input, okay this is the kind of error message compiler should give, because compiler should not be generating code for incorrect programs, okay, it should be able to test both for correct input and incorrect input, okay, so we have a test suite and what happens is that you have test suite of program where you have expected behavior of each of the programs which is documented and this test suite is given to the compiler writer, saying test suite compiler again distribute, okay, and all the test programs must be compiling using the compiler and all technicians must be reported back to the compiler writer, and

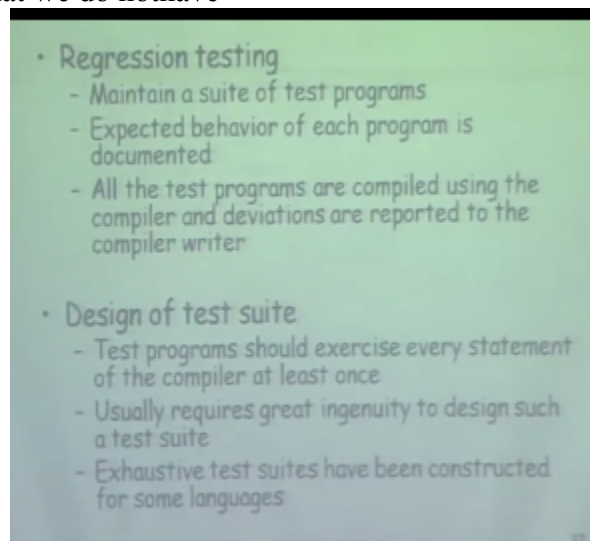
- Regression testing
 - Maintain a suite of test programs
 - Expected behavior of each program is documented
 - All the test programs are compiled using the compiler and deviations are reported to the compiler writer

normally this is not done by compiler writer themselves, this is done by Quality Assurance team, okay, and there is a classical sort of observation in industry that you cannot find bugs in your own program, you sort of trust it all the time so it is someone else's job really, QA team is always a different team which you say I am going to just dissect it and find as many bugs as possible and there use this test suites, so test suites are shared by this best.

Now how do I go through this testing, okay suppose I give you 50 thousand programs, I'll give you a test suite which contains 50 thousand program, and you start testing, okay, now what happens, you start test so suddenly you find that 5999 programs are working correctly, you come to 6000 program and it crashes gives me an error that your compiler now has a bug, what do you do in such a situation? You have to debug your compiler right, now when you debug your compiler probability that you have removed that bug which was not compiling 6000 program or

program number 6000 is gone, now compile, but what warranty do you have that other programs which you already tested they will not compile, there is no guarantee, right, in fact I'm in the classical again, situation is every time you remove a bug perhaps you can use still move, and that is what you will notice in the last week of April, when you start checking your project code in the week of 15, everyone will come for demo (Sir, abhitakkaamchalrahathaabhiabhi crash hogaya, abhi) because I was trying to remove some bug it was working fine, but now it doesn't work anymore, okay, this is half the teams will give me this reason in that week particularly, and you'll notice it now, okay.

So what we have to do therefore is whenever we're trying to remove bug, okay, you have to go through something known as regression testing and what is this regression testing? That whenever you find that there is a bug your program does not work, you fix that and start from the beginning, okay, and go all over once again till you have compiled all the programs in the test suite in a single go, when a single question of the compiler and have observed whatever is the correct behavior, whatever is the documented behavior of that particular program in the test suite, okay. So this is also known as regression testing that you keep on doing it again and again till no program has or till your compiler for all the programs in the test suite gives me the documented results, okay. And how do we design test suite? Okay, so how do we design test suites, okay, so we want to make sure that we do not have

- 
- Regression testing
 - Maintain a suite of test programs
 - Expected behavior of each program is documented
 - All the test programs are compiled using the compiler and deviations are reported to the compiler writer
 - Design of test suite
 - Test programs should exercise every statement of the compiler at least once
 - Usually requires great ingenuity to design such a test suite
 - Exhaustive test suites have been constructed for some languages

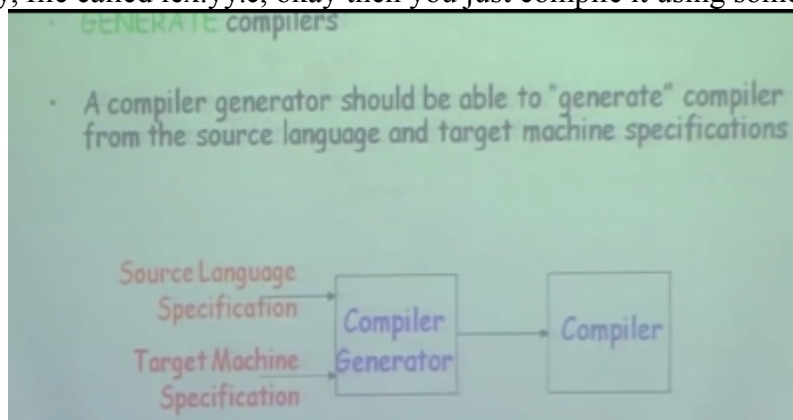
repeatedly program, for example another thing you will notice that when you start doing your project, will write a program and if I ask you how many programs did you use to test your compiler, this five programs, okay, good, so show me these five programs every program has the fault with different values. Okay, so when before going, I'm going from 1 to 10, another will be for I'm going from 10 to 50, now these two programs do not give me new information, they are just repetition, they are more of the same, okay.

So you have to make sure that the test programs you have they are going to exercise different parts of the compiler, okay, so you have to do some amount of coverage analysis, you'll have to make sure that most of your features of the compiler, most of the language constructs of your compiler are exercised during the testing off, this testing process, okay, and therefore test program should exercise every statement of compiler at least once and usually it is an art really I mean the people who design test suites, okay, they are experts in that, okay, and there are exhaustive test suites which have been constructed for some languages like data has an exhaustive test suite and if you notice data is perhaps the only language which has a registered

trademark or TMon top ofthat, which is the language of department of defense and they are the one who havedesigned the test suite and you say anyother compiler in this world if you wantto be call it as the compiler must be tested by Department of Defense US, otherwise it is not going to be called anyother duct of compiler, okay. So this kind of exhaustivetest suites do exist, okay and people haveused them really test there compilers, okay, so this is how, you generate lot of confidence into the compiler, okay.

So now how do we reduce all these effort, okay, all the time what we are trying to do is we're trying to do things efficiently, right, so we are saying we have to write a compiler, we have to test it, and we have to do various things and all the time I am trying to reduce this, development andtesting effort, because I want to cut down the time, okay and I want to cut down of course, okay, so how do I do it? So one simple solution is let's notwrite compilers, we don't write compilers, we don't have to do toilets, we don'thave to worry about one sort of thecredit,life will be simpler, but that's reallynot the solution, that's an extremeposition, okay. Now if you don't writecompilers what do I do? I still need compilers here, somebody has to write now, okay so you got a compiler right there, okay, and what we say is let's notworry about writing compilers, this willsomehow have a black box, now imagine a situation here is a blackbox, they tell this black box I have two ports here, and one port on this site I say Pascal, Motorola and what comes out there is the compiler Pascal compiler for Motorola machine, that would be great right. I just put it into two inversely sticks one as language specifications, one as machine specifications and what comes out is the compiler, yes, no? So can we think of black box like this? Wehave already seen these black boxes, you seen code generator somewhere,have you used code generators, yes,no?

Okay, so let's go back CS251, what are the things you'll get in CS251? You have done this course CS251? 355, we have done CS355, okay. So what are the modules seen in CS355? Anyone who remembers CS355 and modules in CS355? Lex, yeah and (0:45:00) somebody who remembers about lex? Anyone from this side who remembers about lex? What was lex? What is the full form of Lex? That is the manual,do you remember reading the manual of lex, what was the title of the manual? Lex, alexical analyzergenerator, and yaccwas yaccanothercompiler to compiler, right, okay, so what we werereally doing was a compiler should get, compiler generator should be able togenerate compiler, okay, and you will see simplerquestions of this what is lex 2? Whatwas input 2 Lex?What was input 2 lex? What did you say? Don't remember, the set of regular expressions, right, specifications, and what was the output? C4, okay, file called lex.yy.c, okay then you just compile it using something, okay, so



similarly I can think of now compilerin return, which you think languagespecification, which will take target machines specifications and then give me acompiler, if I create this box, okay

then I don't have to write compilers, okay. So I can shift whole effort on writing compilers, who writing a compiler generator and then if I can somehow write these specifications then all I need to do is whenever I get a new language, write those language specifications whenever I get the new machine, write those machine specifications and generate the compiler. Good job, right, very quickly I can generate compilers, okay.

Again, this looks like too tough for me to do like he said writing a full compiler and then taking it to various phases, okay, I might not be able to do it in the single step, I might want to do it in multiple steps, similarly I want to do this activity also into multiple steps and saying that I have language specifications let me say that we have specifications of source and target, but if I look at source specifications, what was the specifications I started looking at, when I was compiling languages, I said I must have a set of alphabets then I must know how to tokenize, then I must know how to check the structure, I must know how to check the meaning and so on, okay, so language specification could be given at several levels of abstraction, okay, so various abstractions for into things like lexeme structures, image etcetera and for each component I can

Specifications and Compiler Generator

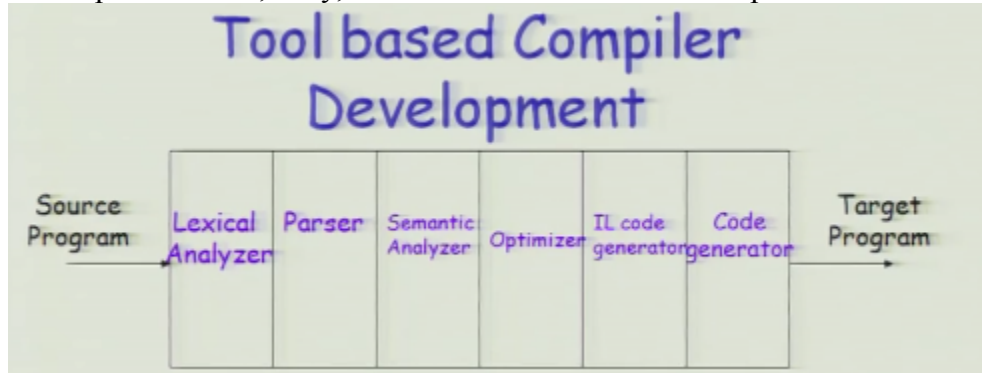
- How to write specifications of the source language and the target machine?
 - Language is broken into sub components like lexemes, structure, semantics etc.
 - Each component can be specified separately. For example, an identifier may be specified as

write separate specification, so when I talk of language specification I can write I can say that when I say something is an identifier I can use specifications in English which says it is a string of characters that has at least one alphabet, starting one alphabet and starts with an alphabet followed by alphanumeric, and they have some sign languages where if you recall your regular expressions I can write something like this, I can say it is the better followed by letter or digit 0 or more than currency tokens, I'm using TV explorer, right and if you say you may have underscore or dollar sign and so on you can again, just make this to this, okay.

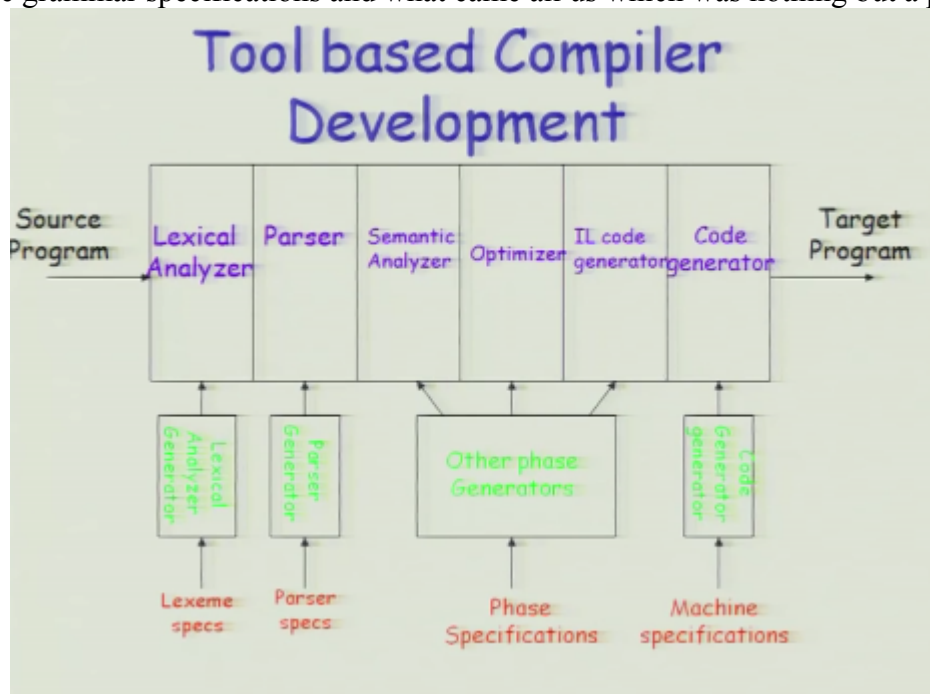
Specifications and Compiler Generator

- How to write specifications of the source language and the target machine?
 - Language is broken into sub components like lexemes, structure, semantics etc.
 - Each component can be specified separately. For example, an identifier may be specified as
 - A string of characters that has at least one alphabet
 - starts with an alphabet followed by alphanumeric
 - letter (letter|digit)*

Now when I am saying I want to write lexical analyzer, this is the only input I should get, I should not worry about more than saying that I use certain data here, I read my input and so on, okay, those are those are mundane tasks I have to do every time only my specification team, okay, so therefore I can similarly write syntax and semantic descriptions and then I can also write target machine specifications, okay, we will see how to write these specification and then



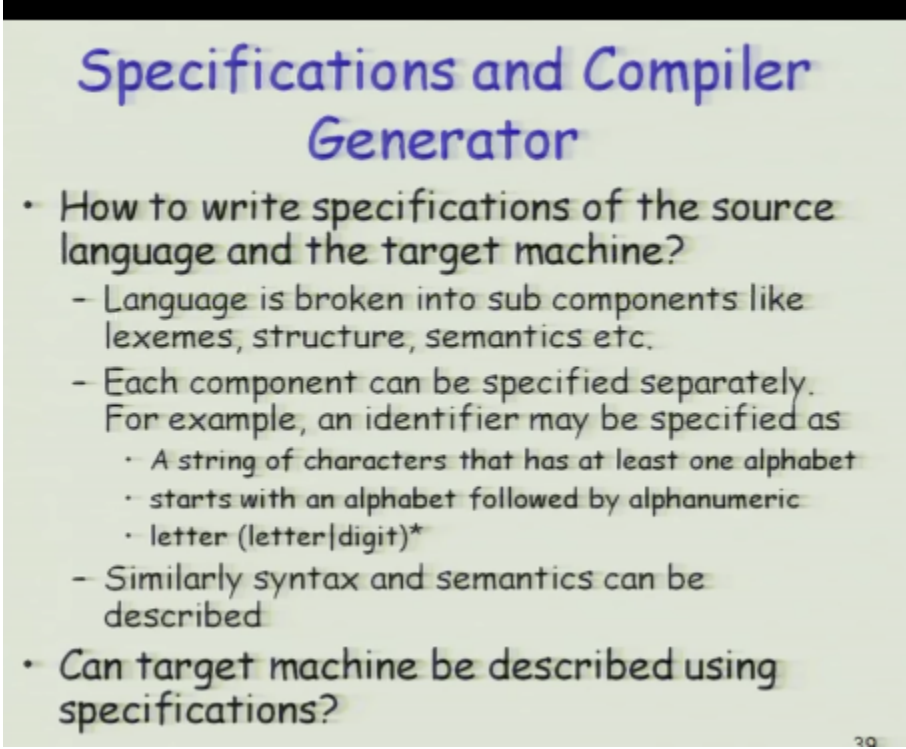
I can go back to my compiler and say if I want to write lexical analyzer, don't worry about writing lexical analyzer, first use a situation, use a tool which is lexical analyzer generator which we have already seen is a lex, and all I need to do is give specifications of the Lexeme's, okay and then what to write, regular expressions for, right, similarly I can have a parser generator, so instead of writing a parser I can have, I can have parser generator and I can write my specifications which are the parser specifications, I can write my specifications in terms of context free grammars, and what is the outcome, this is what we are giving, right, I wrote my context free grammar specifications and what came all us which was nothing but a parser, okay.



So similarly I can think about other phases, and I can have tools for each of the phases, I can have specifications for each of the phases, okay, I can have specifications for code generators, so I can have code generator, generator and I can have machine specifications, right, and what

is the outcome now? I'm not writing these phases but I'm using these tools and I'm writing specifications all the time.

Now will you take life simpler? Definitely as compared to this, the more effort here will be reduced, how much reduced? Okay, so normally we say that if I using certain tools for code generations in compiler we have seen overall effort can go down by almost two third, okay, so it's not that I can do it in one day because writing specification or writing tool is time consuming but if I have total less effort X in writing a compiler by using the tools and using the specifications I can do it in about .65 times which is lot of reduction, and here you are saying 35% effort will be gone and then not just that effort and coding is gone, but your testing becomes easier because now when you say



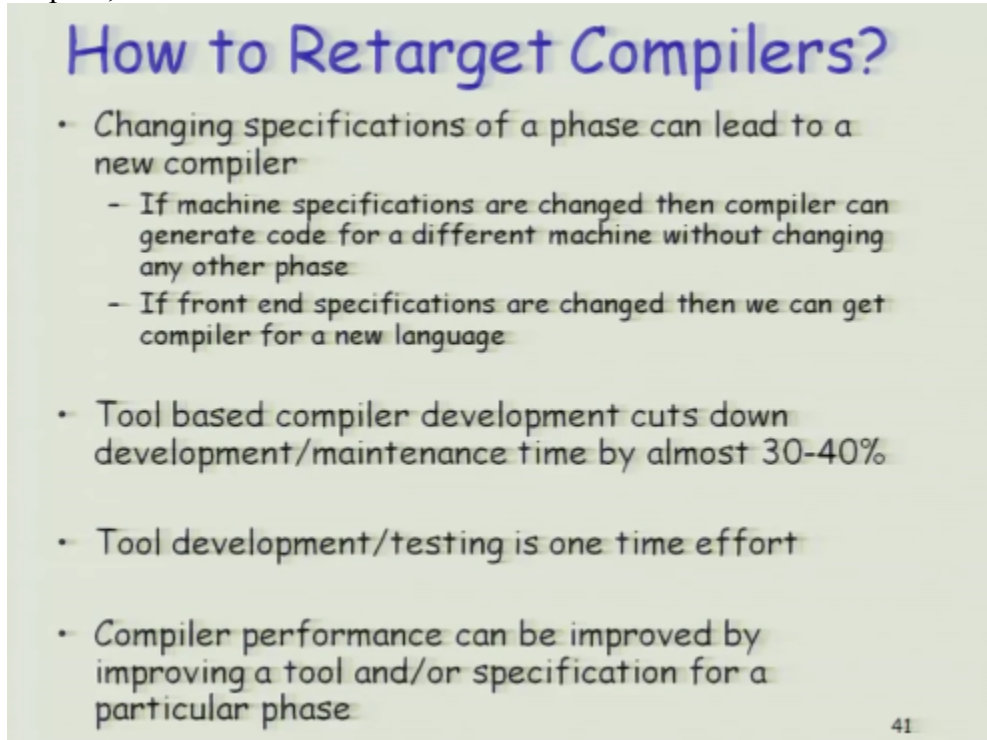
The slide has a title "Specifications and Compiler Generator" in blue text. Below the title is a list of bullet points in black text. The first bullet point is "How to write specifications of the source language and the target machine?". It has three sub-bullets: "Language is broken into sub components like lexemes, structure, semantics etc.", "Each component can be specified separately. For example, an identifier may be specified as", and "Similarly syntax and semantics can be described". The second sub-bullet has three sub-sub-bullets: "A string of characters that has at least one alphabet", "starts with an alphabet followed by alphanumeric", and "letter (letter|digit)*". The second main bullet point is "Can target machine be described using specifications?".

- How to write specifications of the source language and the target machine?
 - Language is broken into sub components like lexemes, structure, semantics etc.
 - Each component can be specified separately. For example, an identifier may be specified as
 - A string of characters that has at least one alphabet
 - starts with an alphabet followed by alphanumeric
 - letter (letter|digit)*
 - Similarly syntax and semantics can be described
- Can target machine be described using specifications?

that I want to give up my lexical analyzer I just need to debug this, I need to say that whether my specification is correct or wrong, I don't have to worry about data structures, I don't have to worry about looping, I don't have to worry about whether I was reading my input correctly or not, that is one thing we have noticed. Generate maximum number of errors, you're reading your input, you either skip a character or read extra character and suddenly you find that something cannot be tokenized, okay, your data structures are not there, all those errors will be gone, they will be part of my tool which I need to test only once again, okay, and this also has certain more advantages because suddenly you find that this is not very efficient because tool is not generating good code, all I need to do is make this efficient, so if you see flex and you remember lex, what is lex? Fast flex, suddenly you found that lex was not giving me very good performance so we go with flex, and then you'll generate the same code, same specification, and you generate much efficient code for the same specification and you have more efficient code, okay.

So each of this phases can be very more efficient just by improving quality of the tools, right, so there is the kind of effort we are going to put in into writing compiler will not look at the code which goes in here but our effort is going to be that what are this tools, and how do I write

specification. This is the error, the focus of this course is going to be, right, so how do we retarget compiler,

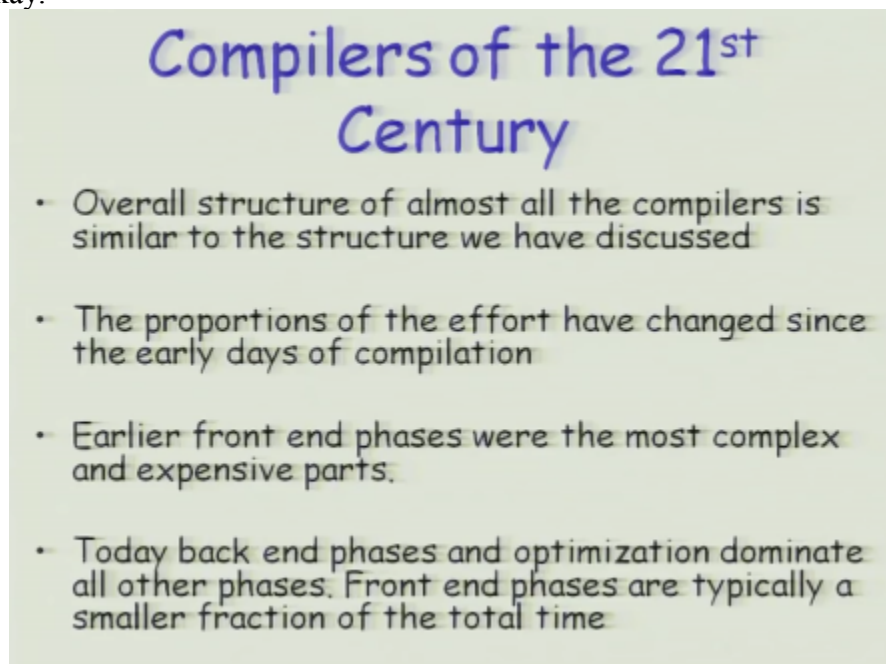


How to Retarget Compilers?

- Changing specifications of a phase can lead to a new compiler
 - If machine specifications are changed then compiler can generate code for a different machine without changing any other phase
 - If front end specifications are changed then we can get compiler for a new language
- Tool based compiler development cuts down development/maintenance time by almost 30-40%
- Tool development/testing is one time effort
- Compiler performance can be improved by improving a tool and/or specification for a particular phase

41

all I need to do is change machine specification, change language specification and all the new compiler, right. So if I need to modify your phase, I'll just need to change specification here, and tool phase and development can cut down your time by almost 30 to 40%, okay, and this is tool testing is only one time of that, okay, and performance can be improved by improving the tool itself, okay.



Compilers of the 21st Century

- Overall structure of almost all the compilers is similar to the structure we have discussed
- The proportions of the effort have changed since the early days of compilation
- Earlier front end phases were the most complex and expensive parts
- Today back end phases and optimization dominate all other phases. Front end phases are typically a smaller fraction of the total time

And this is the last point as far as this introduction is concerned. How do compilers of 21st century look? We are talking about the first compiler of 1957 versus compiler of 2001 okay, we said overall structure is same, but what about the effort, okay, overall structure is still the same but effort is now in that end because lot of optimizations happened, there are different kind of machines we talk about all this multiple machines, we talk about GNUs and various kind of other machines were coming which were not present in backend, okay, so most of the effort has really shifted into the backend and your runtime system, code generation, code optimization has become much more complicated then what we use to be, 50 years back, but frontend is now much simpler, because they are very standard tools, they are good mathematical models for handling the frontends like gradual languages and context free grammars and so on, okay, so total effort has proportion of effort is changed, since early days of compilation and earlier frontend phases was more complex and expensive parts, but today backend phases are the one were most cost is involved and frontend phases are normal expenses, so this is where I close introduction and in the next class we will start on really the need of the phases for compiling, okay, lexical analyzer.

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

DilipTripathi

PadamShukla

ManojShrivastava

Sanjay Mishra

Editing

Ashish Singh

BadalPradhan

Tapobrata Das

ShubhamRawat

Shikha Gupta

Pradeep Kumar

K.K Mishra

Jai Singh

SweetyKanaujia

Aradhana Singh

Sweta

PreetiSachan

AshutoshGairola

DilipKatiyar

Ashutosh Kumar

Light & Sound

Sharwan

Hari Ram

Production Crew

BhadraRao

Puneet Kumar Bajpai

Priyanka Singh

Office

LaltyDutta

Ajay Kanaujia

Shivendra Kumar Tiwari

SaurabhShukla

Direction

Sanjay Pal

Production Manager

Bharat Lal

an IIT Kanpur Production

@ copyright reserved