wyx-uonp-pvi - 2020-08-24

 Dhiraj Jadhav - 08:12

Here morning to all of you. Everyone is good and fine. healthy Now, I think this fear of Corona is now subsiding. Things are started. Our colleges also out of contentment zone now. So let's hope for the best. If College reopen soon, then it will be good for all of us.
Here morning to all of you. Everyone is good and fine. healthy Now, I think this fear of Corona is now subsiding. Things are started. Our colleges also out of contentment zone now. So let's hope for the best. If College reopen soon, then it will be good for all of us.

 Dhiraj Jadhav - 08:12

Yeah, we'll start in minute or two.

S SAKSHI OSWAL - 08:17

 Dhiraj Jadhav - 08:19

Here friends, I think now almost everyone is joined here. Yeah, so let's get started. in any translation job of converting one language to another specifically Computing languages. Excuse me. Some data structures are required. which include of specifically for assembly programming language Okay, let me clear. What is today's agenda. We are actually going to solve two pass assembler. We will input a programming. And assembly program to a two pass assembler. Will create an intermediate code.

That is an output of past one. then That same intermediate code. We will input to pass 2. and then we'll get an output of machine code at the end. That is the whole output of this towards a to pass assembly. So here is one illustrative program that we are going to solve. Now right now we will try pass one. What is required for past one? You will require. machine operation table So what actually machine operation table consists of? Hope this image is visible to you all this image. Okay. So let me insert few shapes here.

 Dhiraj Jadhav - 08:20

this particular portion Yeah. so these are actually a part of MOT table mod table

that means? It consists of related imperative. instructions mnemonics of it that's why it's class is is And the related machine code which is required. in the final output Okay, which is then fed to an income and then 12 order? yeah, so these codes are the final output that we are going to generate and this is the length of each instruction. So if this mnemonic is used. What will be the storage of it? So it is One World length. What are these These are assembly directive?

 Dhiraj Jadhav - 08:39

These are assembly directive. These are also called as pseudo operation table. p o t table Okay pot. The earlier one is m o t table mod. And there is one more DL table. So which consists of declarative mnemonic? what declarative mnemonic it declares a constant or it declares a storage That's why the DS has declarative storage command and DC it declarative constant. Command is present in that table. There is one more table register table. so these register as actually a r e g till creg or drg right and it's classes RG. and there are a number of it machine code for a register is 0 1 machine code for C register is 0 3 and it goes on. The machine code for a d register is 0 4 likewise. So the whole information is treated as a metadata before. creating some more structures out of this one. So what is the output of password and intermediate code? This is an output of past one. symbol table and literal table these three different. Are the outputs of the past one? And then we'll fade this. Will feed this intermediate code symbol table and internal table as an input to pass two. Okay, let's get started. I think a few of you know now Of how this symbol table a little Table app generated because it is discussed in the lab session. Only the students whose batch was on Saturday due to Ganesh chaturthi holiday, we couldn't cover it. Okay. So let's let's revise it for few. It will be a revision for rest of you. It will be the new thing. So let's Let's start so. and assembly program of requires an address from where a program starts so that actually starts with a pseudo operation mnemonic start. And this start consists of an address. from which the location counter is set for each instruction. afterwards so the instruction move are that is next to start will start with location counter 215. That is the meaning. Okay, so let's let's first complete this part. Location counterpart symbol table generation literal table generation and then we'll create an intermediate code out of that. Okay, so don't don't directly you will not directly jump into creating a intermediate code because we require this tables. Okay. So the very first thing what we what will happen symbol and literal table will be generated and then intermediate code and then machine code as the last output of The whole process? Okay, so it starts with 215 now each of this mnemonic belong to some of these meta tables Emoji table pod table DL table register table. Okay, and there is something which is which does not belong to these all meta tables so they might be literals Or symbols. So how literals are identified literals are actually the tokens. Which starts with equal to symbol? Now, what do you mean by token? Now considering space as a delimiter we get. two different entities here one entity is start. Another entity is 215, right? Same here. We get here move our pod table DL table register table. Okay, and there is something which is which does not belong to these all meta tables so they might be literals Or symbols. So how literals are identified literals are actually the tokens. Which starts with equal to symbol? Now, what do you mean by token? Now considering space as a delimiter we get. two different entities here one entity is start. Another entity is 215, right? Same here. We get here move. Our if space is used at the slicer or a delimiter then move our is 100tv get a are easy another entity we get and this comma another entity and the whole is another entity. So NTT is nothing but a token. So actually we are slicing down this whole string file. Now this the whole program consider is as a character stream or a spring file. In which we are going to slice it down to tokens. Now if a token starts with equal to symbol then it is a literal. Okay, these are the constants. so whenever reading tokens encounter with such syntax equal to symbol then. Push this particular syntax or the token into literal table. So your job is to write down either this or identifyin so the little is actually five. The actual little is five. Okay, either you directly push the whole. or You push the number. is coated Okay, so let's let's go with Simplicity. Let's push the whole token here. Yeah, now it is not yet addressed for addressing literal. There is a pseudo. Mnemonic used ldorg that we will see right away. It is ahead defined. Okay, so whenever we we come across this ltrg will explain what this actually do. Okay, just move on. to the next instruction for tokenize it so before tokenizing it first. Increment the location counter by one world, right? Why one word because this move are mnemonic requires One word length. That's why that's why we are incrementing location counter to next. number now this move m a r e g x so X does not belong to a motivable p o d table as well as a register table. And it is also not literal. So what is it? It's a symbol. so insert in into a symbol table That's why we inserted it here yet. It is not addressed. So we'll see where it is addressed where literals are addressed that we are going to see right now. Okay, Moon next so this move M. Also requires one word length. So next. location counter means LC will be 2 1 7. now whenever whenever an instruction starts with a symbol. Okay. So first of all tokenize it first of all tokenize it so you get L1. So what L1 is does not belong to moto DL and register table. It is not starts with it does not start with equal to symbol it starts with an alphabet, right? So if anything starts with an alphabet, and it does not belong to such tables Then it is a symbol, so insert it into a symbol table. Okay. File a reading this instruction if you encountered. symbol

in the beginning itself Then the location counter value will be its address. Again, I repeat. While reading an instruction, if you encounter symbol in the beginning itself, then the location counter value will be its address. So l 1 character 2 1 7 Okay. yet we not completed this reading. Okay. So L1 is done move. Our more is a part of mod table brg. It's a part of register table equal to two. It's a token. Sorry, right? It's a token. What is it? It's a literal. so insert it into literal table Yet it is not addressed. Okay. Yeah, okay Moon next. What will be the next? Location around it now this we come across this. origin, so what this origin is origin is a part of pseudo operation table Okay, it's classes a d means what assembly directive? right Okay, what origin job is whatever instruction next to origin it changes, its location counter. with this value Okay, origin changes the location counter. For the instruction next to it. so Escape instruction location counter value So what L1 is L1 is a symbol what the L1 address is 2 1 7 so do the arithmetic here? 2 1 7 plus 3 how much is it? It is 2:30, right? Sorry. It is 220. Right? So the instruction next to origin will have the location counter to 20. What it origin job is to change. Location counter value for the instruction next to it. Yeah, so we are here now. There is again one more. Just pseudo. of code this mnemonically org So ltur is nothing but LT origin. Literal origin. So what is the origin of all the literals which were inserted into literal table? right now the literals inserted into literal table all are unaddressed Right, so we need to provide some address to it some storage to it. so each of these store one one word, right? So this file so what ltrg job is ltorg To Define ltrg first Define what pool table is so pool table consists of details. What details? available consist of total number of liters Which are addressed? pool table consist of a number which is a total count of addressed literals so initially initially when the program is started the pool this literal table was empty. right and hence all value was also 0 because no literal. Was at rest. Right. That's why 0 so means what currently 0 literals are at rest. Okay. Now this number also serves as an index from which an addressing will happen. Okay, this number 0 will also created as an index. from which an addressing is or will happen. So ltorg will start addressing from 0 index. 0 to the index in what 0 then text in literal table Okay, these two are connected. Will table and literal table? and to hand Okay, so 0 the index pick ya. Hey 5 literal 5 So what will be its address its address will be the location counter value the current location counter value. from where this ltorg started so it is 2 2 0 so it is 2 2 0 Yeah, so l t u r g i say literal address, correct? And no. the number of literals which are an address all get all get addressed Okay in one go. When ltrg is called? So there is one more little interest. So what will happen first it will first increment the location counter. and then link this LC counter value as an address for next literal Now all the literals are addressed right now. So update the pool table value. Now how many literals are addressed? in total in the literal table two literals are addressed right so change into two. So pool table is now consisting the value 2. Okay move ahead. Our job is done of ldrg. Now what next X DS 1. So what this DS is DS belong to declarative language table. DL what this DS means is declarative storage. So this This allocate or this allowed. these many number of words to this symbol What symbol X symbol? So X will be allotted 1 World memory. Okay, but what will be the location counter for this instruction? Simply the next? Location counter value. What is it? 222? Okay, so this instruction says allowed one world memory to the symbol X Which base addresses 2 to 2?

insert this Base address into symbol table. Now this part is also done now currently everything seems complete. Only our reading is incomplete, right the symbol table. All symbols are addressed literal table. All literals are addressed, right? What will be the next location order value to two three? Okay. move our crg, so this is Part of mod table. This is part of register table crg C register. And this is literal, so insert little into little table. simply remove this and add end here.

 Dhiraj Jadhav - 08:41

Yeah. and at the end this is currently now on a test, right so to address. this such literals We need ltrg command. If it is end of the program then end of the program also does ltrg at the end. Okay. So what is the LC counter 2 to 4? Now when and pseudo mnemonic is called it first directly jumps to literal table.

Okay, it jumps to pool table first.

What is the current index from which addressing must be started two? and 10 is present at what index 2 right so it will first check. Is there any incomplete thing into literal table? Yes. So this is incomplete. We must address this 10. So the location counter value related to end. Will be assigned to 10. And now how many literals in total are addressed? There are total three literals intolerable address. So change the pool table value. now questions please questions

 ADITYA PANGAONKAR - 08:41

So there are two Undeclared veterans at the end, then we'll give it two to four and two to five or is there something else?

 Dhiraj Jadhav - 08:41

I connect two to four and two to four each literal will get a unique address.

 ADITYA PANGAONKAR - 08:42

Okay, and a second out was if it was X DS2? So we are located to. Word memory to X. So will that affect anything in the location counter or

 Dhiraj Jadhav - 08:42

Yes, correct.

 ADITYA PANGAONKAR - 08:42

that

 Dhiraj Jadhav - 08:42

Okay. Okay. It's a nice question. XD is 2 let's start with this. So what does it mean X is allotted with 2 word memory which Base address is 222. So 222 and 223 will be part of X. So the next location counter will be 2 to 4. right now

 ADITYA PANGAONKAR - 08:42

right

 Dhiraj Jadhav - 08:42

Yeah, and hence the onward addresses will be updated. in that same

 ADITYA PANGAONKAR - 08:42

here

 Dhiraj Jadhav - 08:42
Okay with one word length.

 ADITYA PANGAONKAR - 08:42
right

 Dhiraj Jadhav - 08:42
So it is a to-do. Why?

 ADITYA PANGAONKAR - 08:42
Yes, so, thank you.

 Dhiraj Jadhav - 08:43
Yeah any other question?

 Dhiraj Jadhav - 08:54
Okay, if there are no questions, then we are ready with. Symbol table and literal table. No next job of past one is to create intermediate code. Now let's start with that intermediate code is a part of is actually now looking at these metadata tables what those metadata tables are mod table machine operation table. then tot table pseudo operation table then I'm talking in sequencer. This red box is machine operation table mot this blue box is pseudo operation table pot. This is a DL table declarative language table. Then this is register table and each of these mnemonics are assigned with some class so if it's a part of a multi table, then it's classes is If it's a pseudo operation table, then it's classes. Assembly directive ad if it's a degree language table, then it's classes DL if it's a resistor then RG. Okay. Now we require this and the Machine code related machine code for the mnemonics which were used in the programming language. Let's get started. So this start belong to which table start belong to. Pseudo operation table. What is this? What is the class of it a d right? The class of it is what a d so a d start with this class comma the machine of code. So sorry the machine code for that particular. Mnemonic, so what is the machine code for start? It is 0 1 so 0 1 then any of this instruction? Okay after mnemonic. There is an expectation of two opponents. Okay. But here is only one operand after mnemonic. It is expected that there will be two operatives and the middle one is and the middle one is resistor. So are there any resistor here? No so Dash. Okay. Check this instruction. After this mnemonic, there is a register. And then there is something else. Then this mnemonic move, are there either register and there is something else. Here move M. There is a register and there is something else right? So same way that is why we put here Dash. which representing no resistor in between so The class is c for it. Or symbol table. The class is s for literal table. The class is L. Okay. Just remember this. symbol table s We try table l. And if there is a constant in between right start consist of a constant. Let's see. Okay. so C comma that particular constant value next move R is a part of machine operation table here it is present. What is the class is? What is the machine? code for a register. It's 0 1. Then there is a little. So if there is a literal occurred directly jump to literal table and check at what Index this literally is present. So literate table so constant is sorry. The class is l At what index and 0 index? This little five is present in two literal table. Okay, so let me complete this if you have still any doubt while I'm doing this, it will be cleared at the end. Let me complete this first. Move M. Belong to machine operation table it's class is is and we're related machine code is what? 5 right and there is a resistor. Each number of resistor the first number of register. That's why we took this one here. We wrote here one. Yeah, then what is there a symbol if it is a symbol and say a con this class will be s and what Index this x is stored into symbol table. At 0 the index, right? Here in the symbol table X is stored at 0 index. so 0 next we encountered in the beginning itself a symbol so symbol. Is L1 stored at wood index industry wall table 1. And what next in mnemonic move are so it belongs to my S. At what index? Sorry at what? What is the machine code of mubar? It's 04. in a register Which number of resistors second number of resistor? Then there is a literal. At what Index this two? means stored in literal table at index 1 next the origin. Okay. this the origin and equ it's intermediate code is not generated. I write here something. origin and equ which intermediate code? not generated Okay. Okay more ltrg. so LP org is a part of assembly directive and it does internally something what it does it internally. Declares a constant, right? So it calls internally DC. clear constant literal is a constant. Okay, it assigns an address to it. What is the length of the address one word? Okay, internally, it calls DC. So let me write This one ldorg belongs to pseudo operation table. Class is 80 and the related machine code is 05. What next ltrg internally calls what DC this DC belong to declarative language table. It's machine code is 0 to yeah. What next? It does not include any register, right? No, resistor so Dash. And what next? What literal? is addressed that literal is mentioned here So what literal this five little is addressed first, right, correct? So what this 5 is it's a constant. What the number is 5? Let me complete the next one. It will be clear to you. Ltrg belong to what? Pseudo operation table class is 80 machine code is 5. What internally does it declare a constant? So BC command is fired. The class is DL. The machine code is 2. Then is any register is used. No, so that's why Dash. And what literal is is addressed by this is two. So this two itself is a constant and that's why we mentioned here that particular leader two. next it's X DS. Yeah, what does X is is a symbol? and what location 0 0 right then BS BS is a declarative. So b l and what is the machine code 0 1 Right and register. No. Then there is a constant. Right. It's 2. It's a constant. So wherever you see Dash. It's a resistor.

 Dhiraj Jadhav - 08:55
If no dash then a specific register is. mentioned into intermediate code Moon next move are passes is machine code is 0 4. And there is a resistor. Which number 3? Then there is a literal. L at what index 0 to 10 is stored at 0 to write.

 Dhiraj Jadhav - 08:55
And what? is ad Yeah. What number? two What internally? Did it BC right? It assigned at the end and addressed to literal. So DC it did DC declarative. sorry nuclear constant so DC Ka

S You - 08:55
answer if the literal table is already generated then why do we need integrated core for lto RG because it assigns address to the results, right? But yeah, this is already ascending the past one. So why do we need

 Dhiraj Jadhav - 08:55
class

S You - 08:56
for past two?

 Dhiraj Jadhav - 08:56
machine code is 0 to

S You - 08:56
kisses Yes.

 Dhiraj Jadhav - 08:58
What is the part of this pool table? Okay. This pool table is used. that yeah, sorry. any question You can you can type into a chat box as well.

B BALAJI PADAMWAR - 08:58
Sir, don't forget

 Dhiraj Jadhav - 08:58
I

B BALAJI PADAMWAR - 08:58
for the end.

 Dhiraj Jadhav - 08:59
register for the admins Okay here yeah. Yeah, please mention it, correct. Correct mention it here, right? Yes. Good observation. Thank you.

T TEJAS PRADHAN - 08:59
If we have some literals which are
addressed at the end then.

 Dhiraj Jadhav - 08:59
of LTR at the end.

T TEJAS PRADHAN - 08:59
follow like the into your J after the instruction.

 Dhiraj Jadhav - 08:59
Correct.

T TEJAS PRADHAN - 08:59
So as many

 Dhiraj Jadhav - 08:59
Correct. I will give you an example

T TEJAS PRADHAN - 08:59
address we will

 Dhiraj Jadhav - 08:59
in the time is up, but I let me give you that.

T TEJAS PRADHAN - 08:59
interventions instructions at the end as well.

 Dhiraj Jadhav - 09:00
first just hold on for a second.