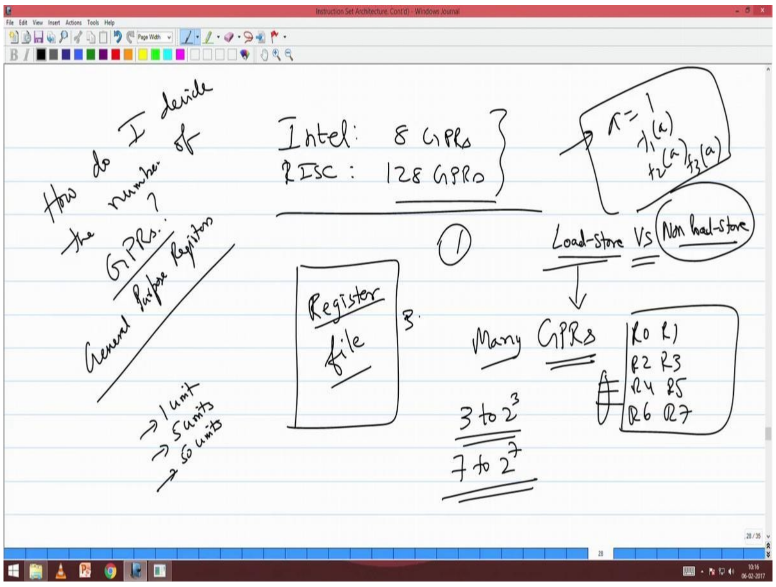
Computer Organization and Architecture
Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 15
General Purpose Registers

Now, let us go and start with some other important topic in designing of the instruction set; how do you decide the number of registers, right.

(Refer Slide Time: 00:31)



For example Intel the architecture you have seen has 8 general purpose registers, but there are risk architectures which have 128 general purpose registers right the number of. So, this is I how do I decide this is the question that we like to answer how do I decide the number of GPRS; GPRS essentially stands for general purpose registers you had seen an architecture in the previous semester where you use only 2 registers, correct, it also worked, right. So, let us now debate like I just want to have a discussion now on what are the factors that will influence the number of general purpose registers; what are the fact.

Student: The basic operations that we could do because we could not do it now we are able to do division because we have EDX EAX and;

So, the first question here is that you are already seen what is a load store architecture verses a normal non load store architecture what is a load store architecture?

Student: (Refer Time: 02:00).

Only to instruction can touch memory one is load another is store all the other instruction work with registers if that is the case then we will need lot of GPRS. So, in a load store architecture we will need lot of GPRS many of the risk instruction architectures are load store architectures right why should risk instructions be load store why should the risk ISA be load store architectures because they are of fixed length almost all the instruction should be of fixed length. So, I cannot afford to have memory operands along with registers if they are constant length I should know where the destination is where the source operands are my decoding should be easy. So, if I have a load store architecture all these things become easy. So, many of the or almost all or all the risk architectures are indeed load store architectures.

The moment I have load store architectures then I need to have lot of general purpose registers because only 2 operations can use memory as an operand the remaining things have to use register as operands right. So, that is one first part. So, there is a need for risk architecture because my execution will become faster my decoding will become faster the moment I have risk architecture they need fixed length. So, I would not like to have too many types of operands there and. So, to make my decoding easy I go through to the load store architecture and the moment I have load store architecture I need lot of GPRS. So, this is one story right now what will happen if I have lot of GPRS. So, let me just take the next point here. So, a GPRS; all the GPRS are stored in something called a register file in advance course in computer architecture we will teach you how to design register files right.

So, there are many registers in this register file. So, when will this register file be accessed whenever I want to fetch data or whenever I want to store data? So, there are 5 stages in the execution of instruction fetch the instruction in which register file will not be accessed decode the instruction fetch data at the pointer register file will be accessed execute the instruction again register file will not accessed while I want to store back the data again register file will be accessed. So, at 2 points the register file will be accessed and when I want to access the register file I should basically say which register I want to read or write from in a case of data fetch I am going to read in the case of writing reason I am going to write.

So, I should tell which register I need to read or write from. So, what I do I give a address for that register. So, let are be eight registers say R 0, R 1, R 2, R 3, R 4, R 5, R 6, R 7, I have 3 bit address for this register this 3 bits are basically decoded to find out which register I should enable right. So, register has always enabled bit then only you can go and read or write from it. So, which register to enable will be done through this decode bit correct write. So, there is a decoder right now if I keep increasing this. So, this is now what 3 2 2 power 3 decoder right.

If I have a 128 GPR it actually becomes 7 to 2 power 7 decoder right as I keep increasing my number of GPRS the decoding will become more and more complex right. So, what it what I mean to go and access that register file again access means read or write access is a common word used for read or writing in to any resource what do I mean by accessing the register file I need to give the address of the register and then I have to go and access it after giving the address there is a decoding happening which basically tells which is the register which is the register that you want to go and access to it goes and enables that register and the decoding happens from a k to 2 power k decoder. So, the total time to access the register includes this decoding time are you able to follow right. So, if I keep increasing the number of GPRS what will happen this decoding time will increase slightly, but it will be logarithmic, but it is it will increase, but very importantly the area of the decoder will exponentially increase right.

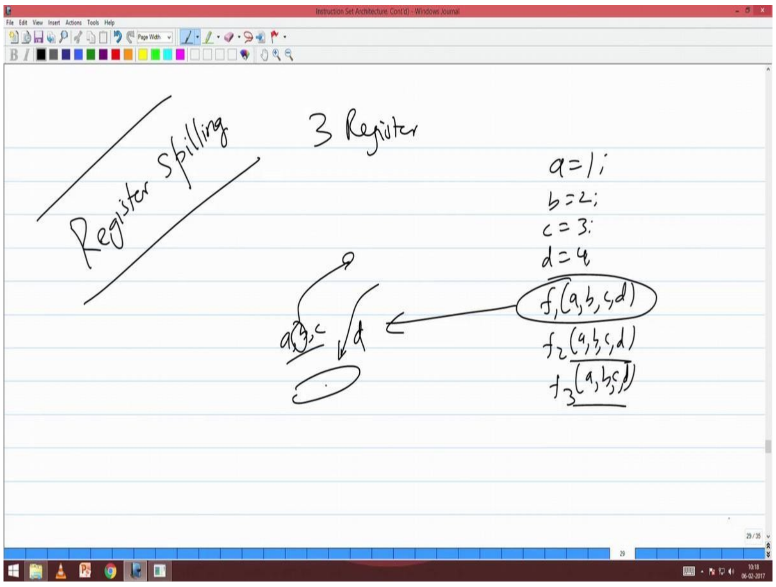
Because I need 2 power k lines right. So, the there is an exponential increase in the total area that is required to implement the decoder. So, as I keep increasing my registers number of registers what could happen I will land up with a circuit with consuming more power more area and also less performance right because the depth of the circuit also increases right. So, this is the other side implementation side problem with number of GPRS right. So, there are plus and minus of how many GPRS should I include right.

So, if I include lot of GPRS let it be there then lot of power area is getting wasted and your performance also will take a hit, but in the same time we have very less number of GPRS what will happen suppose I am computing with say ten variables right if the variable is in the register then I consume say one unit of time if it is in the cache I approximately consume 5 units of time if it is in the memory I consume close to fifty units of time if a variable is in the register its very quick for me to access if it is in the cache it is 5 times slower if it is in the memory main memory it is fifty times slower correct.

So, what will happen is if I have lot of operands which I need to operate upon lot of variables the compiler would prefer to have it in the registers all the variables that need to be used in the near future the compiler will try to have it in the registers because quickly I can keep accessing it and after I finally, say suppose I am having something like a equal to one and there are many many f 1 of a is computed f 2 of a is computed f 3 of a is computed and then finally, you know at this point.

So, right from this point a equal to one till I finish computing f 3 of a the compiler would like to keep that variable a in some register after that it will go and update in to the memory after end of this you can go and say move that register in to memory or store that register in to memory by doing this what happens every time I want to access a for computing f 1 f 2 and f 3 that a will be available in the register and I will get it very fast if it is not there in the register then what will happen I have to go to memory which will become 5 times or even fifty times slower depending on whether it is in the cash or not are you getting this are you able to follow. Now if I have lot of variables and very less number of registers then what will happen is at some point of time I do not have enough registers. So, I will be forced the compiler will be forced to move the variable back to memory for example, suppose I had only 3 registers.

(Refer Slide Time: 10:25)



Suppose I have only 3 registers, but I have I have say something like a equal to one b equal to 2 c equal to 3 d equal to 4 I want to compute f 1 a b c d f 2 a b c d some function f 3 a b c d I can store only a b c. So, when I am computing even f 1 I will first compute part of a b and c and assume it Is a low storage architecture right I will I will first compute a b c keeping them in register then I will have to remove of some b out to memory and bring d from the memory and then finish the remaining part of the calculation right and then similarly when I am doing f 2 or f 3 every time I need to do this I do not have enough registers to maintain my temporary my variables there.

So, as a result of it I go and write that some variables to the memory and fetch it back whenever it is necessary this is this is called register spilling register spilling is the is the is the name for this; this phenomena where I have limited GPRS general purpose registers I have more variables that I would love to store as in the general purpose registers because I am going to use it in the near future, but I do not have enough registers. So, what happens is at regular intervals I have to go and save some of these registers in to memory; that means, the corresponding variables I am storing back in to memory and fetching that new variable from memory to do these computations.

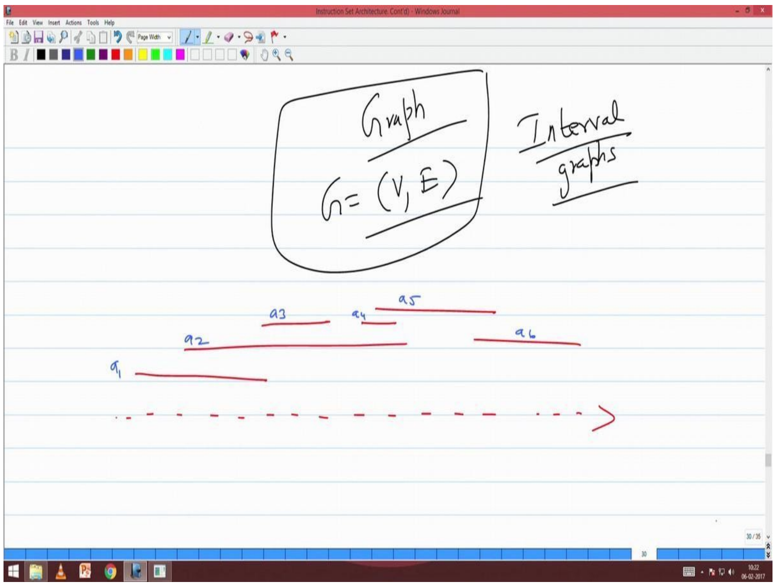
And this becomes very important if I have a load store architecture because in normal operations like add subtract division multiplication I cannot use I cannot use general purpose registers I cannot use memory operands. So, every time I want to do any computation then I have to put that operand in to a register and if I do not have enough registers then we land up with what is called as register spilling and because of register spilling what will happen your performance will degrade because every time I have to go to memory and bring it from memory your compilation also will become extremely difficult because when the compiler is generating for example, the code for a b and c and d it does not know.

So, it has fetched first fetched a b c should I fetch a b c or b c d or a c d or you know b c d I do not know right which combination should I keep and when the new variable has to come which should I evict. So, all these type of things has to be done by the compiler when it is generating the code are you able to follow this, so, that also. So, the compilation also becomes extremely difficult when I have less number of GPRS right.

So, register spilling really kills. So, the objective is now is that, so, as I as I keep on telling you computer science is also computer science if you get something you lose something there is always act of balance and that is why this course becomes interesting right any course in computer science becomes interesting right. So, today I have lot of benefits by having GPRS I have lot of you know problems while maintaining this GPRS right. So, how do we straight that balance I cannot have too much I cannot have too small. So, what is it why should I have 128 should I not have 64 can I not have 256. So, how do you come out with this decision and that is very very important and the ultimate objective is that I should not overdo it nor should I land up in a situation where I am having register spilling correct is it fine.

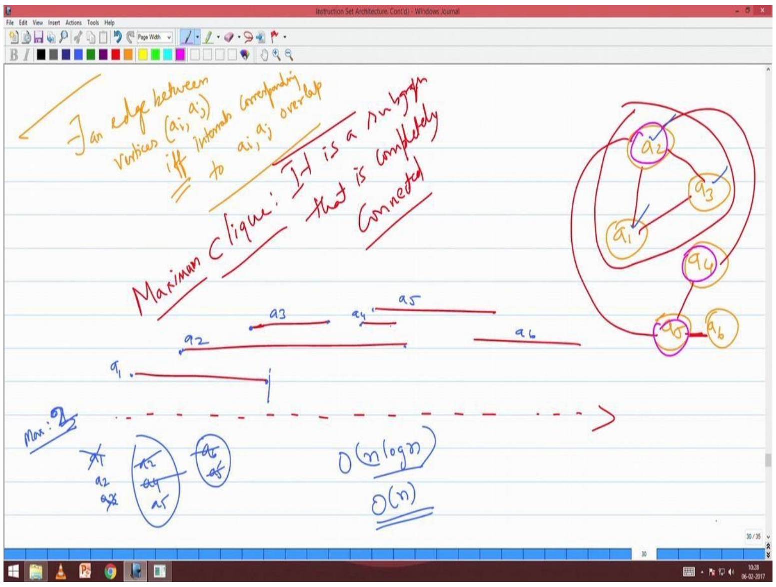
So, how will we solve this problem? So, I am going to go. So, somewhere I am trying to connect algorithms graph theory and all here. So, normal computer organisation course does not teach these things, but I will tech you these things because these are very very important and crucial because how architecture is designed using some of your theoretical [FL]. So, let us go back. So, let us now start what is a graph pala gap tell me what is a graph collection of vertices.

(Refer Slide Time: 15:05)



So, graph g is a tuple V comma E where V is a set of vertices and E is a set of edges connected to that now let me introduce a new type of graph called interval graphs. So, let this be the number line or this be the; this is a number line. So, I draw several intervals each interval in the line would be some starting point and an ending point. So, how many intervals are there in this stuff there is a 1 a 2 a 3 a 4 a 5 a 6; there are 6 intervals now I construct a graph using these 6 intervals where in every interval represents a vertices vertex every interval represents a vertex.

(Refer Slide Time: 16:31)


Instruction Set Architecture. Cont'd) - Windows Journal
∃ an edge between vertices (a_i, a_j) iff intervals corresponding to a_i, a_j overlap
Maximum Clique: It is a subgraph that is completely connected
a1
a2
a3
a4
a5
a6
Max: 2
~~a1~~
a2
~~a3~~
~~a2~~
~~a4~~
a5
~~a6~~
~~a5~~
$O(n \log n)$
$O(n)$
30 / 35

So, how many vertices will be there in my graph 6 vertices a 1 a 1 a 2 a 3 a 4 a 5 a 6; now there exist an edge between any vertex I and j if the corresponding intervals overlap. So, there exist an edge between vertices a I comma a j if an only if intervals corresponding to a I comma a j overlap now a 1 a 2 there is an edge a 1 a 3 also there is an edge then after that no other edge from a 1 a 2 a 3 there is an edge a 2 a 4 there is an edge a 2 a 5 also there is an edge a 3 to a 2 only there is an edge nothing a 4 a 5 there is an edge and a 5 a 6 there is an edge let me define a variable a term called clique what is a clique of a graph it is a it is a sub graph that is completely connected.

It is a sub graph that is completely connected I want to have the maximum clique here the maximum click of this graph is 3 right a 2 a 1 a 2 a 3 is connected take right you cannot find 4 vertices a set everyone is adjacent to every other correct and I cannot. So, the maximum size clique is 3. So, I leave it as a very simple exercise what you need to do is you will sort all the intervals in terms of this n points starting point and ending point in terms of the starting point and then just go do a traversal from left to right. So, what will happen a one is added now what is the max clique I have seen. So, far one right then a 2 start point, so, this is called left point and right point for every interval a ones left point as comes. So, a one is added a 2s left has come it is added. So, max clique is

2 now a 3 is left point is come then it is added. So, max clique is 3 then what happens a

ones right point has come. So, a one you remove. So, max clique still remains 3 I have only 2 fellows here.

Now, now after that a 3s last point has come. So, a 3 is also removed now the number of element is one, but max clique remains 3, now a 4s first point has come. So, a 2 a 4 there are 2, but still and now a 5 has come say a 2 a 4 a 5 again max clique becomes 3 a 5 has come now a 4 gets removed then a 2 gets removed then a 6 comes. So, a 6 a 5 is there then a 5 gets removed and a 6 gets removed. So, when I scan from left to right and keep adding something into the basket when I see a left point and remove them from the basket when I see a right point, I can go and find out what is the size of the maximum clique, right. So, if there are n intervals to swap them in some order takes n log n time after that to find the maximum clique it takes just order 2 n time or order n time. So, the point is given an interval graph, I can go and find the size of the maximum clique in order n time a order n log n time where n is the number of intervals; are you able to get this yes or no yes no.

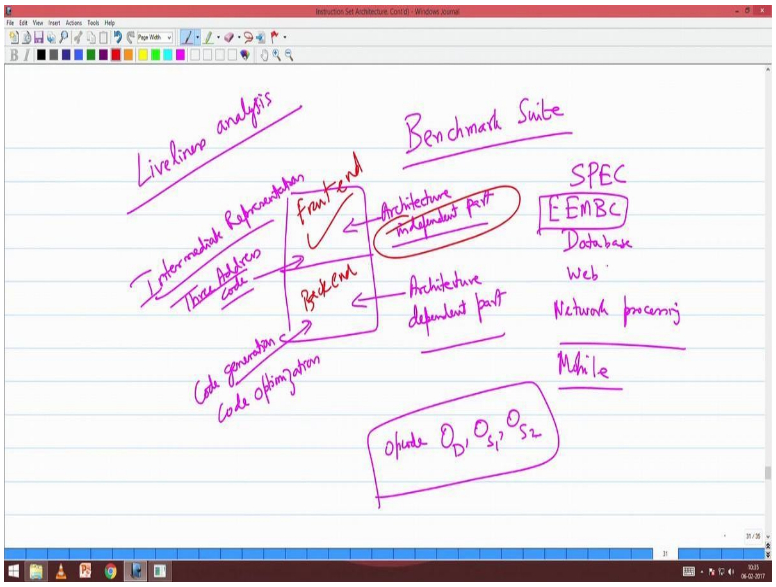
Student: (Refer Time: 21:42).

Yes. So, there are 2 3 cliques here a 2 a 1 and a 3 and a 2 a 4 and a 5 that also we forgot I thought a 2 a 4 and a 5 is also a triangle in the sense they are adjacent.

Now, why are we learning this in the context of register spilling now what will happen suppose you are going to design architecture, now you are expected to go and run some programs on that. So, this architecture need not be a general purpose computer it can be an high performance computing it can be a computer for doing data base it can be a a data base transactions it can be a computer which is trying to do a web base transaction it can be a computer just doing some digital signal processing it can be a mobile phone it can be a base man computer it can be a computer that needs to sit in a router. So, we could have several types of processor today network processor embedded processor you know control programmable logic controller class processor micro controller class processor desktop class processor server class processor what more we can keep on distinguishing more.

For every type of processing today we have some work load work load is the type of program that we will execute a collection of such type of programs basically is called a bench mark suit suppose I want to design a server class machine today right.

(Refer Slide Time: 21:23)



Instruction Set Architecture, Cont'd) - Windows Journal
Liveliness analysis
Benchmark Suite
Intermediate Representation
Three Address code
Frontend
Backend
Architecture independent part
Architecture dependent part
Code generation
Code optimization
SPEC
EEMBC
Database
Web
Network processing
Mobile
Opcode O_D, O_{S1}, O_{S2}

Let us say web server then there are bench marks suits which will be representative programs that will run on your architecture these are available these are available at a huge cost this is a big business selling bench mark today is a business. So, there are many bench mark suits that are available let me just name some of them you go to Google and find out more about them spec is a bench mark EMBC is a EEBMC is a bench mark there are data base bench marks there are web based bench marks there are bench mark for network processing what is a; it is called bench mark suit why we call it as suit it will be collection of programs which are representative programs that will be running on a system we have mobile bench marks today now when I am designing an architecture for say some field x to solve some problem x for example, I want to do something for micro controllers micro controllers that sit in your washing machine your things etc right simple micro controllers micro controller that even control your nuclear reactor all small and big things.

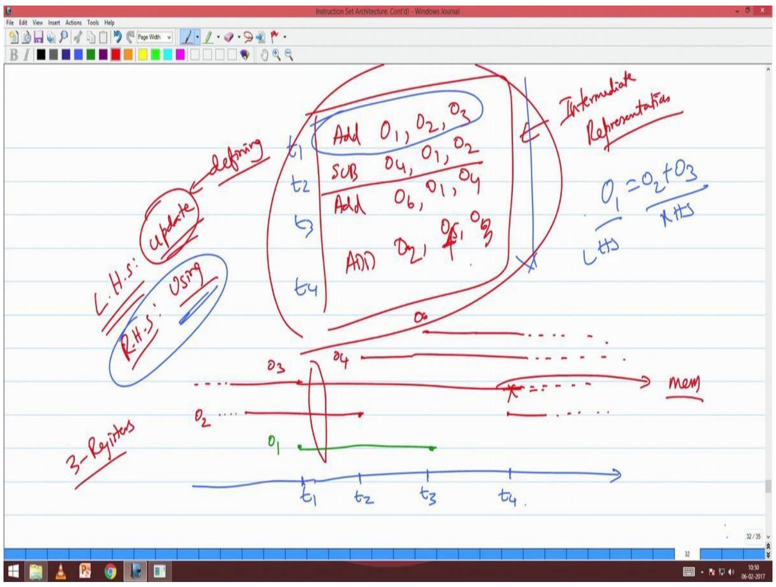
Now, there are some embedded bench marks EMBC actually gives you a set of embedded bench marks. So, you take these bench mark programs right and do what we call as a liveliness analysis what do you mean by a liveliness analysis now let us go back to compilers I want to teach some very interesting fact about compilers; compilers are in 2 parts.

So, when the compiler a single compiler comprises 2 parts one is what we call as architecture independent part another is architecture dependent part architecture independent part and architecture dependent part in the architecture dependent part you do all your code optimisation first code generation and code optimisation that is you translate the code in to the machine language of your target architecture suppose it is x 86 I translate in to a x 86 assembly programs right while the architecture independent part takes your c code or whatever high level code and make it into what we call as a 3 address code this is basically an intermediate representation a 3 address code is nothing, but one opcode like add and etcetera.

And 3 operands operand d operand s one operand s 2 this is how a 3 address code looks like where operand d is destination operand s one and s 2 are source operands. So, your entire program is translated in to an intermediate representation which is a 3 address code in which 3 or 2 2 3 or less than 3 operands and the maximum size is I will have a destination operand to source operands. So, this is basically a 3 address code now this 3 address code is basically translated to what to the corresponding machine code now this type of constructing a compiler is extremely useful for designing the next generation architecture because the first part when I compile I can compile it on any machine and that is independent of your architecture.

So, I am trying to design architecture. So, I need a compiler which will give me something which is independent of this architecture. So, this 2 stage construction of the compiler what some time they call it as this is the front end and this is the back end this 2 stage construction of the compiler is extremely useful for me to design my architecture correct. So, the front end and the back end. So, what I do I want to design a new architecture I take all? So, suppose I want to do it for say web transactions I just take all the benchmarks representative benchmark for web transaction I compile it using any compiler and I stop it at the first other front end, so, will have the 3 address intermediate representation of all my programs.

(Refer Slide Time: 28:46)



Now, the 3 address in implementation would be something like add R 1 R 2 R 3 subtract R 4 R 1 R 4 R 2 add R 6 R on1e R 2 R 4 and again add R 1 R 5 R 6 now this is an intermediate representation. So, I will just say I will keep it as o right sorry I do not want to confuse it as registers. So, I will say o 1 can you just say R 1 R 2 R 3 right o 1 o 2 o 3 what was sub o 4 o 1 o 2 add next one.

Student: 6.

O 6 o 1 o 4 and o 1;

Student: o 5 o 6

O 5 o 6; these are operands now when I do o 2 o 3 see in an instruction whatever is there on the left hand side is the updated is the variable which I am updating right and what essentially I am I also call it as I am defining the value of that variable whatever is on the right hand side I am only using that value right for example, in this add o 1 o 2 o 3 I used o 2 and o 3 this is nothing, but o 1 equal to o 2 plus o 3 this is the right hand side and this is the left hand side I use the variables on the left hand side I used please note here I use the variable on the left hand side to define or update the variable on the right hand side. So, o 1 is valid it comes to life whenever it is defined and that life or that avatar of o 1 is valid till when till it again gets updated you are getting my point. So, you have variable or a operand actually comes to life when it is updated or defined that is it is in the first entry of these 3 address code or when I model it as an equation like o 1 equal to o 2 plus o 3 it is on the left hand side of that equation and it is life time is there till it does not die or its new avatar starts only after when it is defined next.

And in this avatar it need not be useful always till it is used it is it is you know it is it is alive after you stop using you can go and redefine it later, but if you stop using at some point then it is almost dead. So, let us take this. So, there are 4 instructions let us call it as t 1 t 2 t 3 t 4 because the program is going to execute in this order let us say it add is going to get executed in t one then there will be t 2 is t 1 plus 1 t 3 is t 1 plus 2 etcetera. So, everyone one unit of time it is going to. So, let us draw the time line here. So, for o 1 it will start with t one and end with t 3 because t 4 again it is going to get defined.

Now the next avatar of o 1 will start here and it can go wherever it wants right similarly o 2 right just to make that thing little bit interesting let us make this o 2 o 1 started at t 1 and it is valid till t 3 after that o 1 is not used assume o 1 is not going to be used similarly o 2 started somewhere and it stopped at t 2 now and again it is going to start at t 4 and probably it will go further. So, o 3 started somewhere and it stopped let us say o 3 was here till t 1 o 3 was here and let me say that this is also 3 it is going to be used till t 4 because o 3 is used till t 4. So, it started somewhere and its going on and on and o 4 maybe there it is not there for add o 4 again starts at t 2 and let me say this is o 4 t 2 and it goes on o 4.

And o 6 actually starts at t 3 right now at any point of time if I can maintain all the live variables in registers I am done if I can maintain all the live variables in registers that is enough for me right my program my program going to execute fast. So, for example, at t one what are all the live variables o 1 o 2 o 3 all the 3 are live variables if all the o 1 o 2 and o 3 are available in register then please note that t one the add operation will go very fast when I go to t 2 please note that o 1 o 2 o 3 and o 4 right are live variables if I have all of them in registers I can do that thing very fast no sorry yeah; o 3 is still there right now in t 3 again.

Now I see o 1 o 3 o 4 o 6 and so, on if I have 4 registers I can do the whole thing very fast right there are 5 variables here o 1 o 2 o 3 o 5 o 4 and o 6, but if I have 4 registers I can do it very fast suppose I had only 3 registers what would have happened since the load store architecture o 1 o 2 o 3 will be on registers then I want next when I want o 4 o 1 and o 2 o 3 should have been sent to when I come to this particular instruction o 3 should have been sent to where memory I should have moved o 3 to memory and then I can use o 1 o 2 and o 4 correct. So, this is example of a register spill are you able to follow this because when I move of o 3 to memory at this point again I need o 3 at t 4 I need o 3 here.

So, I have to bring it back from memory are you able to follow right. So, if I do not have 4 registers here then there is a certainty that there will be a registers spill if I have 4 registers this particular snapshot can be done with all the variables maintain in registers and which is necessary right in a load store architecture it is a must that the all the variables of non load store operations are in registers right. So, how do you calculate? So, what is this representing this is again a set of intervals I can take the; I can construct a graph here from this interval right and the size of a clique is equal to the; at any point of time.

The size of overlap the number of overlapping intervals represents the size of the clique the size of the maximum clique tells me the maximum number of variables that are going to be live at any point in this program right the size of the maximum clique of this interval graph size of the maximum clique of this interval graph basically tells me the maximum number of variables that are going to be live at any point in this program does it now right. So, now, given this interval diagram now I can construct an interval graph as we did in the previous part and on this sorry where it I can construct an interval graph and from that I can find the maximum clique and that maximum kick will tell me for this benchmark program this is the maximum number of variables that will be alive at any point of time and. So, if I have that many numbers of registers I will I can execute that program without any register spill correct.

So, what you do I take the entire set of benchmark programs I take the entire set of benchmark programs and in these benchmark programs I do the liveliness analysis this is called a liveliness analysis how long a particular variable is alive that is what we are doing right in the in the in the here how long a particular. So, I do this liveliness analysis construct interval graphs find the maximum clique of these interval graphs and the maximum of those maximum cliques size of the would be the number of general purpose registers I would like to have.

Now, if that is too high then we will strike a balance with a caution that yes for some classes of benchmark programs even within benchmark I could have verities of benchmark program some classes of benchmark programs this will have memory spilling and it will take some more time you able to follow. So, this is how we go and design on the maximum number of registers while making architecture are you getting this. So, let us very quickly I am looking at load store architectures load store architecture I need to have every operation done on registers I cannot afford to have memory spilling compilers have 2 phases a front end face and back end face I am designing the architecture. So, there cannot be a back end for my compiler. So, that is those this type of a compiler constructions helps us.

Now we go and look at the front end of that compiler we get what we call as a 3 address code from the 3 address code we go and do a liveliness analysis we construct an interval graph find the maximum clique of this interval graph and that will tell me for all the benchmarks and that will tell me maximum number of GPRS I need to have this is. So, this is one very nice example where graph theory is used conception graph theory are used for constructing architecture I just want to cover this in this course because you get a link to the other courses that you are study. So, when you go and sit in some math course or graph theory course do not think its waste of time for computer science its very very useful for computer science right any doubts yes you had a doubt.

Student: Sir what are the valid operations in analysis like add and subtract you had mentioned add and subtract.

Anything it may see as long as see with the variable I can do as long as I am getting defined whether it is true add div sub or some operation which you and me defined some queues multiply and add something like that as long as I my value changes that variable becomes alive till it is keep it is being used somewhere you stop using it then you will say goodbye.

So, in the compiler course the first part of your compiler course probably one month or even lesser than month you will learn what is a 3 address code the 3 address code will have some generic set of operations and a structure as I have shown you in this place right this; this type of a structure right and from that you do and you can decide because this does not need an architecture you can use c compiler for and. So, the front end is independent of architecture.

So, I just use a c compiler and just say do the front end part alone. So, I do not need to use a c compiler for x 86 architecture or anything this is this is architecture independent I stop at the middle I cut the compiler into 2 halves. So, let us take even a x 86 compiler or an arm compiler no problem I will remove of the bottom portion then both the compiler are same normally how do we construct c compilers you have one compiler with machine independent part architecture independent part then you have an architecture dependent part.

So, the front end of all g c c should be the same the back end only will change. So, I take just a front end and generate this 3 address code and do this analysis. So, where do you compile this program you can compile you can run the compiler on any x 86 or any debug machine as long as I get the 3 address codes after that I do the analysis on that same thing debug machine finally, from that I will find out how many GPRS are necessary for my machine correct is it.