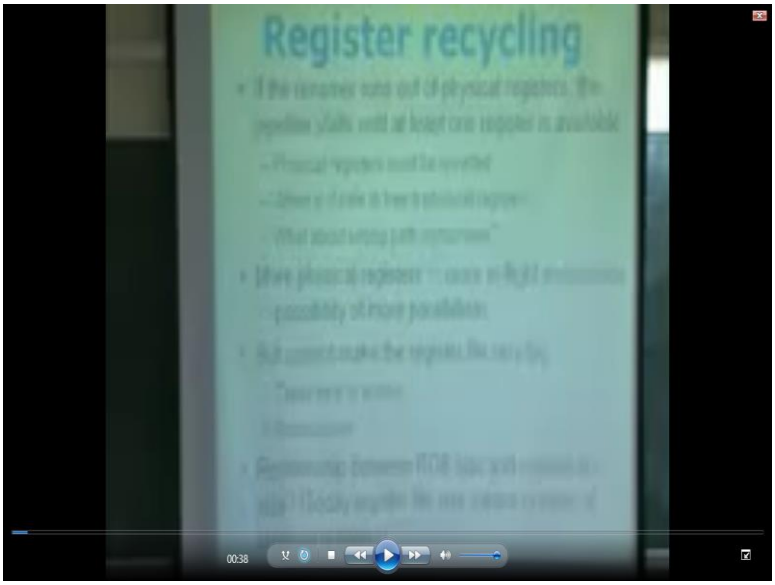
Computer Architecture
Prof. Mainak Chaudhury
Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur

Lecture - 24
Dynamic scheduling, speculative execution

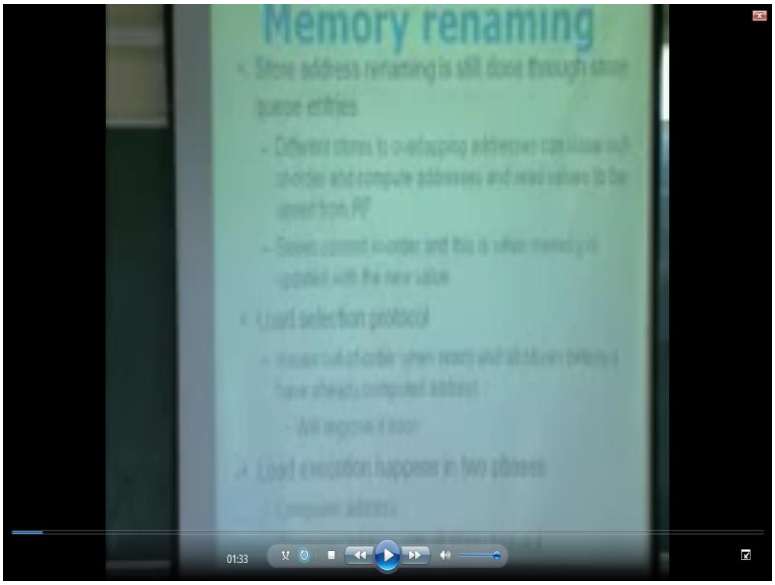
So, last time we are discussed register renaming in detail and how the registers. So, the final conclusion was that the reorder size.

(Refer Slide Time: 00:34)



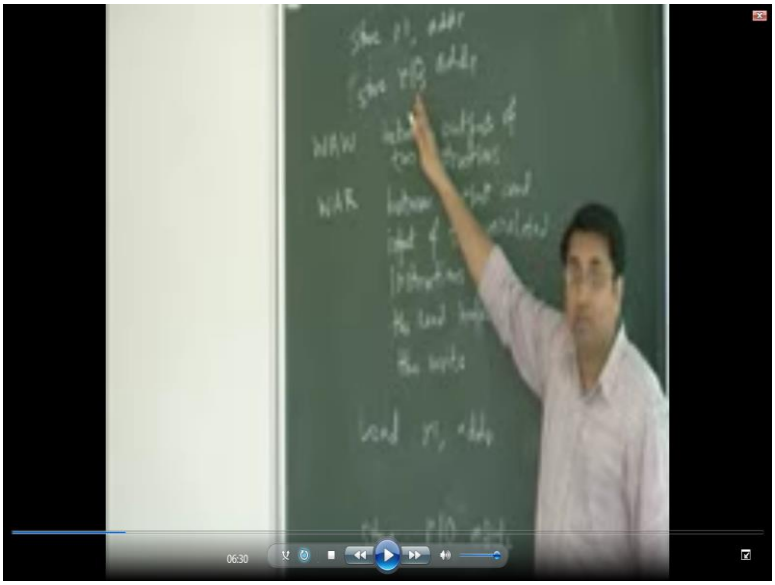
The reorder was a cheapest chapter from maintaining the order of instructions. And a straight final instructions register recycling we also reduced relationship between size and register file. And confusion was that if here more physical registers if we have more in flight instructions that allowing extracting more parallelism. And to support of course, fundamental assumption here is that to accommodate in flight instructions we have to and to have in require this will not physical registers relationship.

(Refer Slide Time: 01:33)



So a 1 small thing it was left was about so memory. So, register renaming which is the with the fire when our registers my get picked register instructions and that my interfere with the output of some other country instructions what? We have the right of the written right of the read depends. And these are purely and effect of pilot and that is to remaining we turn master order is password. So, ideally if you have large in off register file you only be review the file you are true need after 9 month. So, memory renaming is a other thing that is which the problem you have there are tolls.

(Refer Slide Time: 02:45)



So 2 situations essentially talking about WAW and WAR, so WAW is basically between outputs of 2 instructions. So, outputs happened to be the same the outputs happened to have the same storage 2 instructions there you have a write after write dependence. And here essentially we were talking about between the output and input of 2 unrelated instructions where the read happens before the write. So, we read for register instruction so second instruction happens to the write to same register simple. Because of the file is finding. So, the memory dependence is exactly say we are talking to 2 dependences says here it will happen to be 2 store instructions. To store instruction I could see memory location how it this 1 not really on audy fact of any complier source is basically program property that 2 store instructions write to same location all right. And that is the and the write of the write assert and such fundamentally is almost it impossible to release.

Because the wrights have to happen to the same location there is no exception. Here we talking about a store call by a load so and the store writes to memory location the sorry a load forward by to store. A load reach memory location x and store write to save memory location x all right. So, again this so it see the problem the problem is we could switch other 2 instructions the load is going for so you have a load some registered r 1 to add j from add j and they after in structure we have to store the same add j. I am aim to maintain this all right the loads have happen before this store there is no exception.

Otherwise the load of the load is getting wrong value from this add j all right there is the WAR assert we will going to know. Now, the question that would to answer is this for is WAW and the war is basically stored of some registered some address then another store after distractions on a some register. So, what the question is there any WAW then it can actually ignored this you could see reason of ignored order of the address the store make it very big to the load.

So, there will be there will be leading space if you thus order all right so that is sacred also this store before this store by the redeem you mean that r 10 becomes available before after 1 all right and here r 10 it was happen before r 1 r 1 plus that may be some single act. So question they do have you could using safe has did the register for we will do inside the prosper we will have some special storage for holding the values of the store in structure called store buffer so store. I just remaining still down in this so we that we hour sorry we distribute issue to across coastal areas to had an litigable we have you had the store and load queue all right. So, the store queues increase are little special in says that may have a will or store will the value and a such you need such a, because the store way is any time. But we could not update comes to the all right. So, now, we execute this 2 stores out of war of ignoring this it has the values a such will be copied from in this copied from r 1 to its particular store queue at the value here will copied from r 10 to this stores and ultimate.

Of course, they will memory go to the advantage here is that we are able to over lack address computation of these 2 instructions over lack of again you can it will have in half resources and whichever execute any these 2 of enable to you can send. So you can compute the address and we can greeter registered files out of out of put the values in the stores. Of course, eventual the memory of it will happen similarly; here I can use this store growing this load completely all right.

And put the value in the in the store prevention of this particular store how react careful handle this to load eventual? So that must be taking care of so this is so here the hole process is summarized so store at that remaining it still down store currencies different stores to can sure about and compute addresses and keep values to the store from the registered files all right. Here I mention particular over clocking the address, because that

is the problem that is where dependence that is the all over of course, you can execute whatever you want stores commit in order and this is when memory is updated with the new value. So, memory update is still, what are the load selection protocol load issues out of order when ready and all stores before it have already computed address so these have already discussed.

(Refer Slide Time: 09:50)

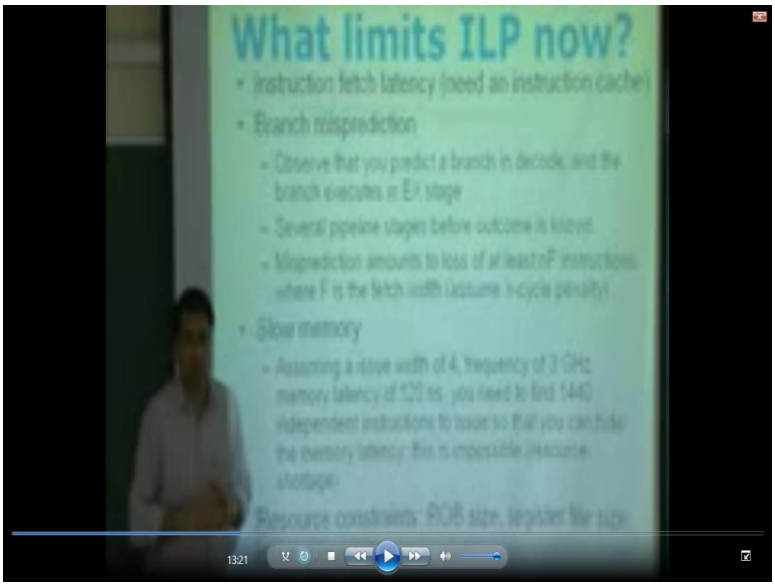
So we have you had out of between loads and stores you can imagine a paused loads to queue as a all right so have the load instruction. So, what do we saying is that we can issue the load instruction whenever you want whenever the offer load instructions already how there have to is do wrong thing that you have to check? If all the stores report to this store have already issued on and by issuing already issued and total with address. So that that has to be guaranty in, because renamed this load will issue we will check if it is address hole address any of the stores address.

And if you does we were discussed in the last class that so depending on the hole time if you by either take a value from the from the longest store weak of the load or we can you can let the load weight until the complete. So, this one look a really nice will soon improve it, because simple reason is that there should be a mechanism to not make for the all source of file. Because this load may not depend all the stores dependent the war

the loads all right. So, this is the very approach to just be correct main loads so we will improve upon this very soon.

And load execution happens in 2 phases as we discussed earlier first the load will issuer. First of all this criteria has to be has to be satisfied for local issue when the load ultimately issues it will first compute the address. And then it compute the address with a outsource. So of course, to know this criteria satisfied not this has to be happen first to load has to compute that so the why it is balance that whenever the ready I will issue. It all right it will compute address comeback check all the stores here all the addresses and then if there is a match it will do something you says no match. So, once we achieve effective views that we are recruit concurrency between the memory installments and done by the memory renaming to the stores the entries that is all. So, temporary in cal nation of this particular address this becomes once story temporary as on as processor this store ignores achieve signify this particular address.

(Refer Slide Time: 12:37)



So after doing all these things for to be signed in half so number 1 is instruction fetch latency. If you have to growth the memory brain the instruction recycle that so we need an instruction cache. The problem of caches after this instruction cache is should hold depending on the management policy of the cache most and hopefully you can use than

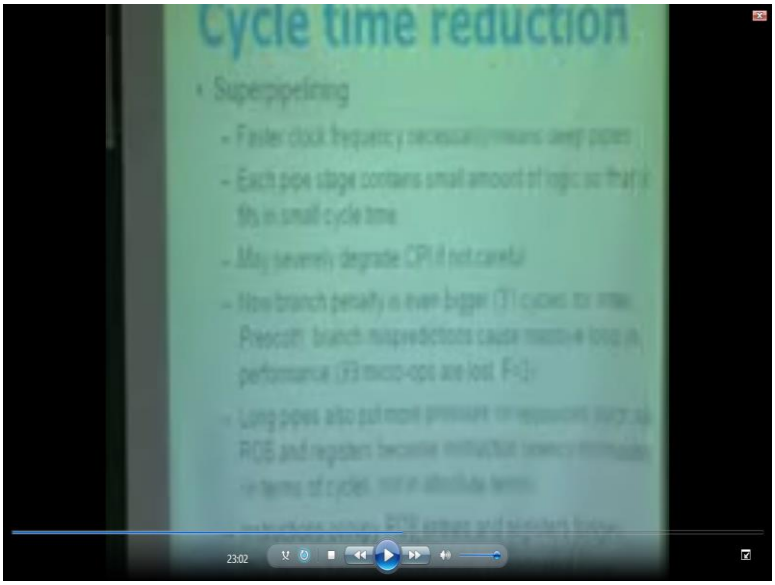
to the time. Second problem is branch miss predictions so observe that you predict a branch in decode and the branch executes in ex stage of the pipeline. So, there were several pipeline stages before outcome is known. So, miss prediction amounts to loss of at least ((Refer Time 13:25)) instructions where f is the fetch width assume n cycle penalty. So, that we give out if every cycle facing f instructions, because in you can miss prediction either any number of warms at the full so that the huge loss actually. And a and the if you so if you are going to deep file last fluctuate the implication is that you should have very good branch benefit you read the benefit of. On 1 is slow memory just too just to give a feel about a this problem here is 1 particular beta points, nothing to re particular password just to remind to point the problem here.

Let us assume that you will issue width of 4 be leave that maximum full stop assume refer. Because here is that may one let we see 120 nano second. So there the question is suppose 1 will be operation this currently in fact all right, so particular memory loss that low use. So, to be able to height the latency of this particular load operation how instructions your need so that could heights this native. This passes I should it should look like as if there is no operation. So, there will be once announce able times timing look the processer should read into is to 4 structure every side just a find somehow four instructions at the side take all right what is amount to 120 nano second is 365 actual regard all right multiply by the 4 1440 instructions. If load instructions outstanding in the case let us see must be able to find 1440 independent instructions to issue. So, fundamental requirement to able do that I need to be fall at least otherwise, so a essential these impossible, so just a possible attempt.

The main problem is not have a resource you cannot support such a and if you want support such a it is a required talk about that miss predict size there is slightly performing minus number of that so discuss last so yes. So, this is what ultimately becomes the problem resource constraints the rob size the register file size and issue queue size. So, these structures ultimately repute ILT and this is related to this particular, so in addition to that we have other 2 forms all right. So, will talk about this particular problem in great detail in the remaining course of accounts how to resolve this issue actually? This is a very hot process problem, because this where to be zone to program and it could be find out that for assigning the programs spends time. You find that 70 to

80 percent or even more I want to time spent memory just accessing memory that is all. So, the data has become extremely slow that is the big problem related to the process give small guide of the process compute very quickly. And ask for the in this you do not have value it of intelligence in system we provide the data that lost that lost. So, this the problem and will discuss the solve the solutions that that will, but till know problem and then all right so any question?

(Refer Slide Time: 17:52)



So, going back to basic little bit call that dissolves or equation for computing execution time statics for instruction time is instruction count time sacrifice. So, for we have focused on the first component that is how to reduce the CPI or if true line IPC instructions retired per cycle. This one is a is the determining by the profile instruction count the architecture a such cannot be much the profile instruction to execute that is it. So, architecture is a person this is small stamp of course, not even instruction sector that we decide of course, the provision reduction to be combined all right. But the micro architecture implementation of privacy as nothing to do instructions that is the concept for it so in problem of thought component now, a little bit so cycle time reduction is another technique to boost performance all right which associate in half faster clock all right. So, you are know and to the edited person should understand the product of pre terms you cannot ignored the first at least generally cannot be the first. And first to the

last chance are actually are not will be there which ever very fast processer officially CPI is very large.

So, here you get and will exact that is for it talking about now that one discussed so how do you get faster clock frequency? There is usual architecture view ignoring the device physics and engineering part there is only 1 way to that is you make your pipe for this war all right. If you other pipeline if you double the frequency you half inch 5 stage you take pipeline if you not no frequency you keep a whenever you can all right let see that you want exam problem how maintain the pipe the frequency? So, that is what architect can do you get faster clock of course, passed you can have smaller promises which can which can faster fast advantages all right. So, let talk about that so these pipeline stage should be one cycle for balanced progress break pipe stages into smaller stages to get faster clock. So what will do about this what is the problem of making pipe bigger can anyone tell?

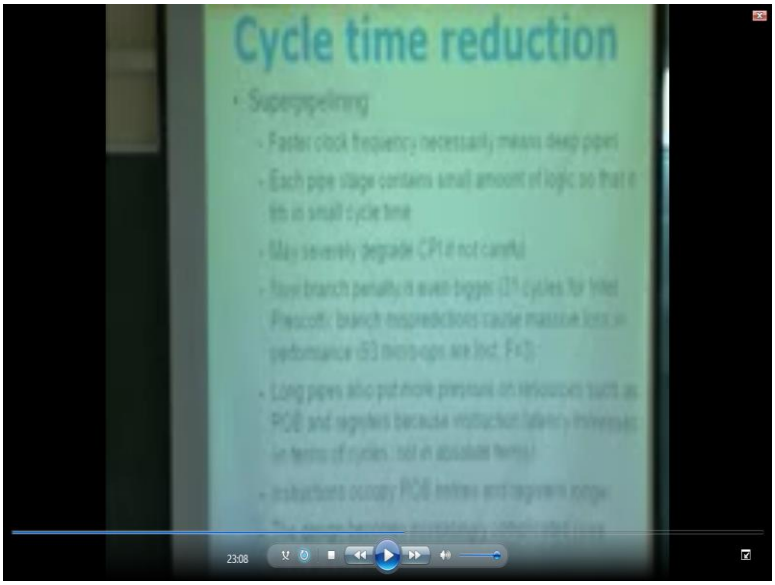
Student: This prediction priority in this anything ales.

That what exactly it is it is priority does not cycles what will absolute time we are talking about that the rejection. So really overview your cycle time we want buy you more execution time all the time all right. Let us by pass fast increase in numbers what is the impact on the point of execution time you not possible this is take time design.

No time no design usually structure.

Fine, let we see increase, because the length of the value make of the passing have tired exactly, so all these second together may actually not review the clock frequency by that however pipe to be frequency pipes is take is grow ahead all right. So, pipe required to, because I will taking ten stage make it which regards, because of this all right. So, a so the we have the couple of the things that we to all the other frequency. And ultimately for it means is that and that particular cycle time whatever target it was you affect CPI you losing the opportunity.

(Refer Slide Time: 23:01)



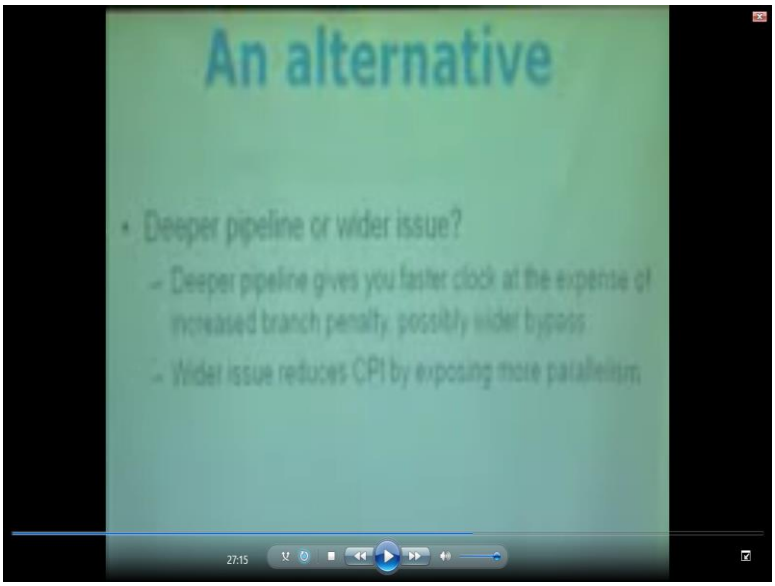
So super pipelining is time that is used by a computer architect to essentially the phenomena that you are pipelining too much all right. And, but it what is used for getting faster clock frequency so each stage contains small amount of logic. So, that you can fit in small cycle time may severely degrade CPI if not careful I just discussed 2 points right you can branch penalty is even bigger you are talk about that. So Intel Prescott was one of the one of the processors which Intel Prescott is the highest papers all right. If we are talking cycle branch so other points as well in the frequency stage as there is many other promise branch miss predictions cause massive loss in performance. For example Prescott had issue with of face width of 3 micro ops pre cycle you could so 31 cycle's loss if essentially mean 93 micro ops are lost. So, every miss predictions you make you lost 93 micro ops is the big loss actually it will touch for 31 cycle along the wrong path and left for path long pipes also put more pressure on resources such as rob and registers.

Because instruction latency increases and some explain this advantages saying that if you have deeper pipes you might put increase your rob size and register files what is that? Exactly so since your deeper pipe you have no instructions in pipe line at a any quantity time of that much share off that increase your instructions. And we already know that rob size and registers size as connect the future if you have larger rob if you want to use the hope the entire of a have a larger register file also. And whiles in a problem well, if you

have a register file your register file as it does not make it. You talked about disaster is smaller faster you make some structure review is going to score. So, here the register file gives that closed now, if you want this target this frequency you have to now, have a deep rectified register file access which it is more.

And essentially now, feedback new right you get the pipe deeper and deeper it ignored the registers you quietly to the time bigger 1.8 why not cannot make it bigger at what essentially what would be reached faster clock frequency. You have to slow of processer that what? So, instruction occupy rob and registers longer the design becomes increasingly complicated wire delay does not scale. So, mentioned that bypass becomes slow means is that the bypass values cannot be very good in time know what ever design talked about saying that while another the value in 1 cycle right for immediately usually happened. It really you pull the bypass you may have to wait for some time just, because bypass passes in wrong and multiplex becomes so while that just a accommodate time so it has many down size of pipeline, so because this is the point it will see it will c p l if you not here.

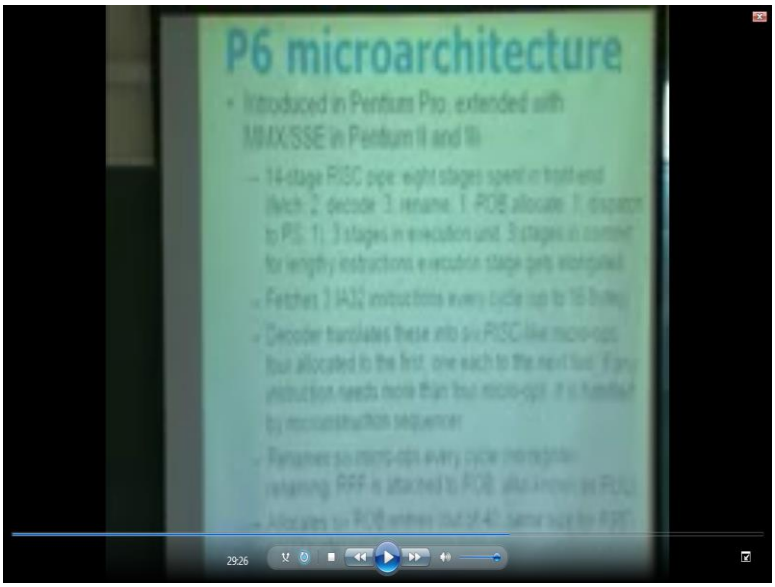
(Refer Slide Time: 27:06)



So get if a what you by do not make my pipeline deeper have a several points what have a wider issue I saying that I have a five stage find and happy with a small frequency. But

I what to do is I issue instructions very wind the other have a we have discussed instruction about cp the problem that is means that we should be find those structure otherwise you waste issue. So deeper pipelines gives a faster clock at express the little transparently possible divided bypass why there issue CPI by excluding more transaction. And which was depends on they will your program till offers so much of that is you have fill up you issues o k s all right so is are there have limit so programs. But usually you want to you find that unable to three infrastructure and what is the other problem having very sorry. And there in a the issues which is not the same it should be instruct so re such a bypass here that is maximum right yes addition to that is into that the substitute the files in their instruction will more. Because the wakeup login to depend on it yes of course, the otherwise you have will be that is what?

(Refer Slide Time: 29:22)



So a review of the p s i x micro architecture so these goes back a long time actually a reason is Pentium pro and extended with a MMX and SSE in Pentium 2 and 3. So what is these accvarent step that one you must I hope must a what is Linux multimedia extention what is assess? I believe everybody is heard assesses who as it sorry no simply first sorry. What it single structure modification that is that is a right a simply yes what is each some tell me it becomes no extension forget extension. Yes you want to find out no restrict screen script so they were introduced page at 2 and page at 3 all right just those

value of necessary all right. So, Pentium pro was the first processer you can offer Pentium a Pentium was super scanner processer with 2 piper we clear about or not. So, we are Pentium pro was the first architecture it introduced the execution and the regards so it was the 14 stage risk pipeline so as a nation you want to exercise i s a with total make prospers to risk micros.

Let it so whichever pipeline looks a there is pipeline eight stages spent in front end. So a makeup 2 cycle fetch 3 cycle decode can anybody guess why decode is so wrong? 2 cycle fetch 3 cycle decode 1 cycle rename 1 cycle rob allocate and 1 cycle dispatch to resolution stages that is 1 thing and then resolution stage infrastructure stage wait for 3 stages in execution unit and 3 stages in commit so that is the forking stage time what is decode are wrong? You need it is rob access rob allocate want to says why decodes is lost no not in the decode size that is the real. Very interesting why worth processer in a home you seen it decode sorry the instruction side is very complicated extremely how? It so smartness not before exercises instruction actually. So that is the reason why decode de code is so complicated yes that is what the instruction for line instructions execution stage gets along get it beyond 3 cycle these the benefits of these stage the fetches 3 i a 32 instruction every cycle who has not see this particular tom. I a 32 what is it what is stand for do not confuse exact sorry they it Intel analysis so fetches 3 i a 32 instructions every cycle a to limit that it can up to 6teen bytes.

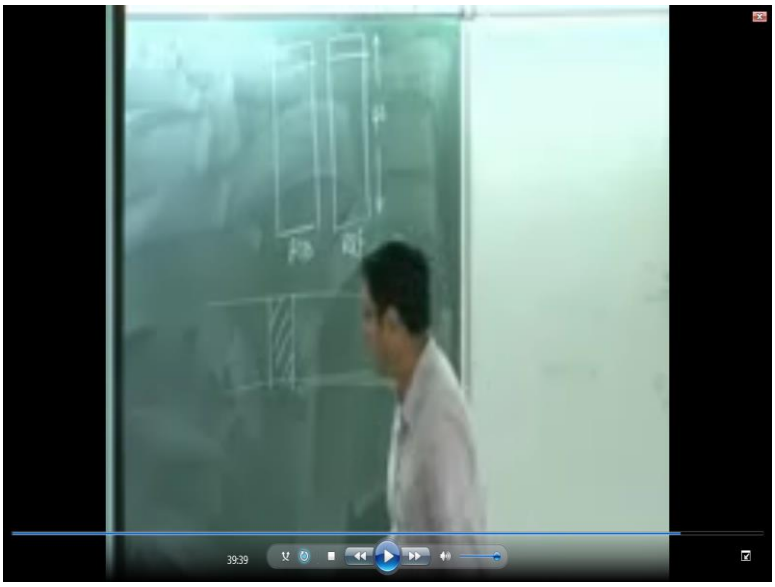
So, essentially what I am saying is that it is the minimum of these 2 3 back three instructions of 16 bytes whichever is minimum that much decode cycle decoder translates these into 6 risk like micro ops of again is the max down to the decoder. If decode maximum of 6 translates into 6 micro ops four allocated to the first one each to the next 2 all right. So the 4 decode into micro ops these a static allocation of instruction to the translates to the micro ops. The first instruction will get four micro ops all right an one micro ops each to the next 2 instruction. If any instruction needs more than 4 micro ops it is handled by the microinstruction sequencer. Because you can see that this is the maximum all right that the decoder can handle translation all right renames 6 micro ops every cycle. And this bandwidth as to match decodes bandwidth make should the pipelines smoothly. Otherwise you know you know in the in the named there rename 6 instructions 6 micro ops then make sure at the manual stack de code stage.

Student: If the second instruction more than 1 micro then it could be actually

Yes so actually what the do is that a the take this 3 instructions so slot the complex instructions of the part a request of the simple instruction. But yes I have what to do, so it just give a idea of how the complex in decode which kind of you have to relax. So, rename 6 micro ops every cycle I interestingly a Pentium pro did not have any register renaming all right. So, it looked like our initial design a simply issued the renaming will should be so. And Pentium pro call this particular this a real slots of the register file essentially is that the rob entry as special fees. That means so that value of instruction value and that particular filed a taking into call that Italian register file.

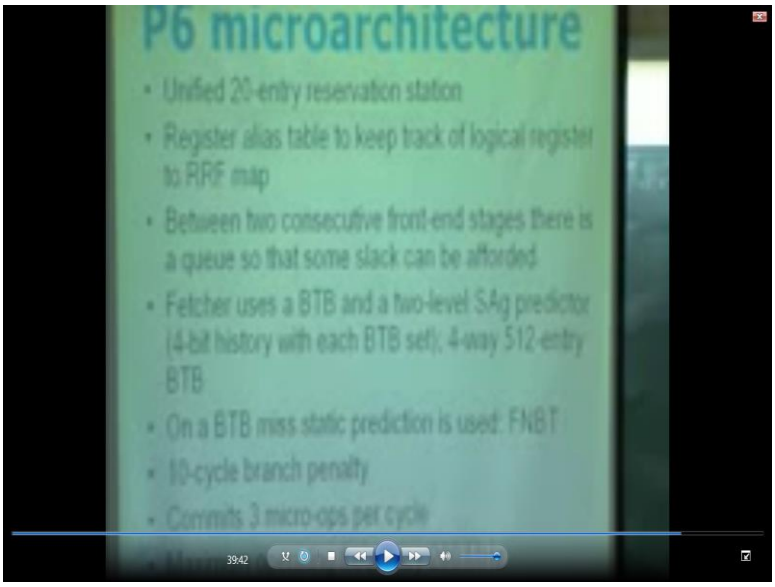
And the main cost of instruction executes what is the value the values put in that that particular filed and renaming that filed until the instruction retires. And that time the value copied from the RRF to the actual instruction that is why rename constant. And will some academic resources it is also called RUU register all those RUU a is actually more complicated rob and we are talking about RRF. If you want read about RUU I can give you and the allocates 6 rob accessories this has match decode the relevant 6 recycle you get RRF every cycle all right. Rob size is 40 this allocates 6 and RRF also forty as we will guess, because RRF essentially free duck in have again to right.

(Refer Slide Time: 39:17)



So a this is the rob a different cycle you can see the particular filed have a extension of this particular all right suppose size stable.

(Refer Slide Time: 39:42)

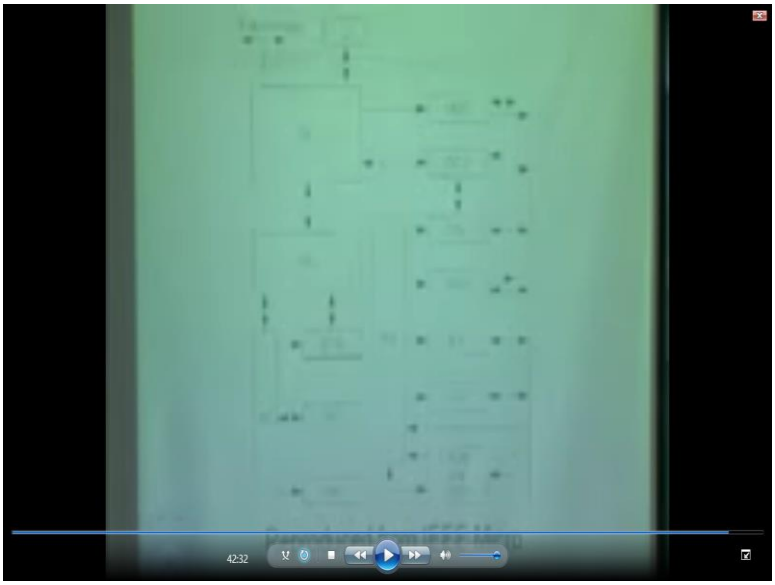


There is unified 20 entry reservation station so a this a vertical formula distributed architecture. And talk about these issued 20 entry issued in is a register alias table to keep track of logical register to RRF map that is like map we talk about between 2 consecutive front end stages there is a queue so that some slack can be afforded. So, that is between stages between decode the a queue of instruction so at the Pentium decode can as usual. So same further renaming stages also for some reason stock what could be eligible value of renewal stall anybody guess decode is not stalled asusal. It means that in the a queue between decoder rename before the campus until for the renaming stall what should be that why renaming stall sorry there is no musical this is rob right make it you are right you correct announce. Rob right the rob that other because these 2 are locate into that right.

So, rob allocate let me back to put into rename and gradually it capacity all right it does not yes there 2 different stages in rob allocate yes effectively it results rob entries at that point. By allocate I mean at this stage right instruction in the rob these are should be already happened here the allocated value of instruction of else in the decode cycle. The

fetcher uses a branch toper buffer and a 2 level sag predictor. So, there is the four bit history of each b t b set and a four way 5 12 b t b so the 10 cycle branch penalty why commits three micro o p s per cycle maximum operating frequency 200 m h z it might's of 1 a.

(Refer Slide Time: 42:32)



That is your this is micro these for is a high level looks like. So this, the instruction the docks to the b t b the instructions to the decoder this is micro instruction sequencer. The register slain of the map that talk to the rob phenomena the reservation station which you should instruction in the execution units. Let see what will we have instruction execution unit the adjustable unit here should be compute the access of the memory. And these a memory instruction unit it stock to the data cash it goes to the and next level of cash all right. And the memory offering buffer it maintain order of instruction to go for to the memory so this side of system. So, I can give the reference you can read more it.