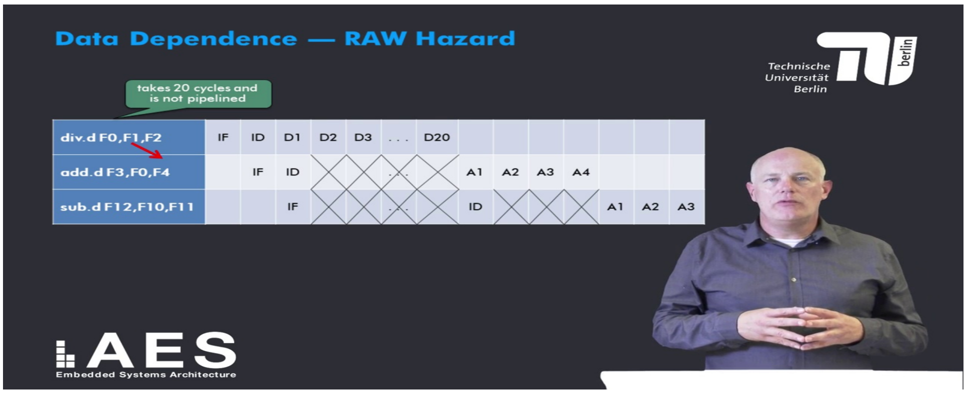
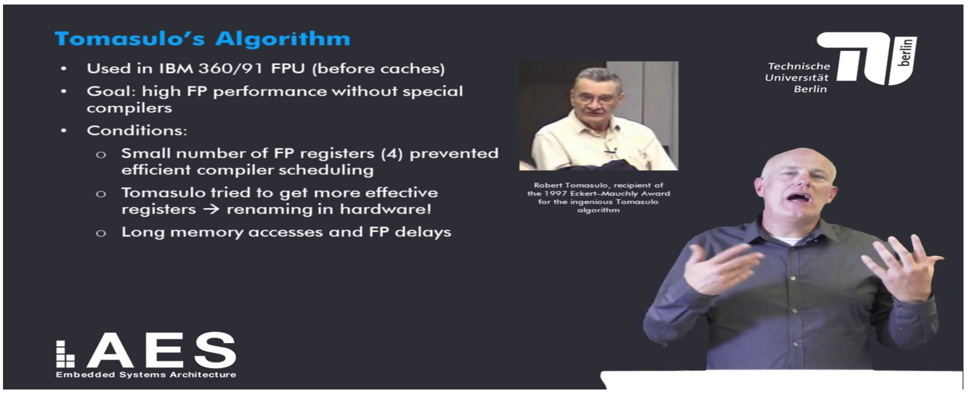
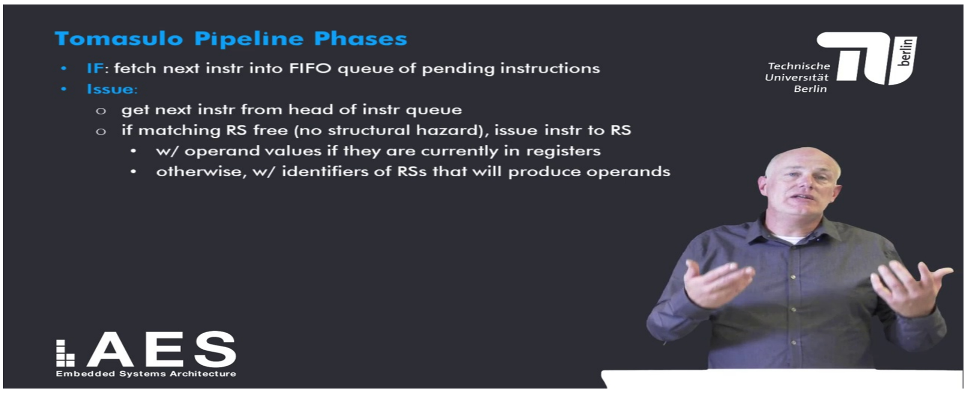


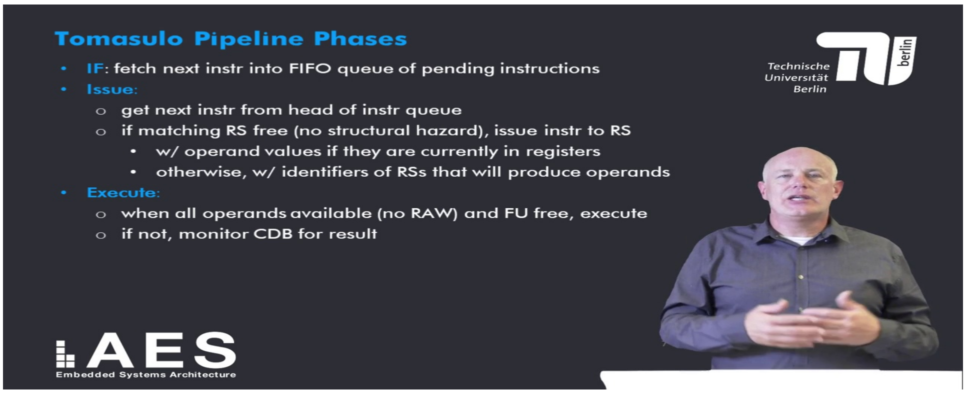
In this lesson I will describe Tomasulo algorithm and how it implements dynamic instruction scheduling. Here are three MIPS instructions that divide double and add double and subtract double. Since the divide writes F0 and the ad reads F0AS indicated by the Red Arrow.As before, I assume that floating point divisions take 20 clock cycles and are not pipelined. The first instruction incurs no stalls because it is the first one.The main point of this example is that the subtract is stalled even though it does not depend on any previous instruction.There are also some stall cycles caused by structural hazard.An analogy can be drawn between dynamic scheduling of instructions and roads and cars. If one car stops all cars behind it have to stop also. If one car stops the cars behind, it can take over. This is a picture of Robert Tomasulo, the man who invented the algorithm in 1997.



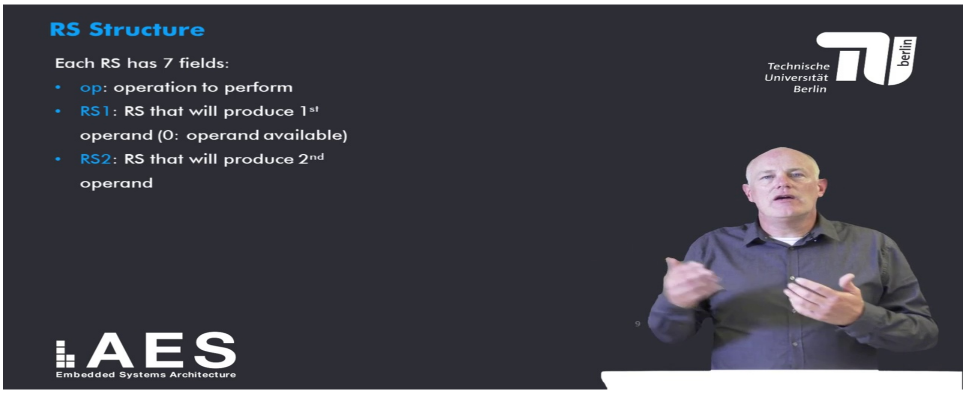
To understand their architecture and organization we need to study this algorithm. Here's the same example as before.To be able to do so, we need to get the add double out of the way, so to speak.Or broadcast on a bus called the common Data bus to all reservation stations. I purposely say phases rather than stages because pipeline phases may take several clock cycles, whereas the pipeline stage always takes a single cycle. In this phase, the next instruction is fetched, but not in a single pipeline register, but into a 5 foot queue of pending instructions. In this phase, the next instruction is retrieved from the instruction queue, and if there is a free matching reservation station, the instruction is placed in that reservation station.



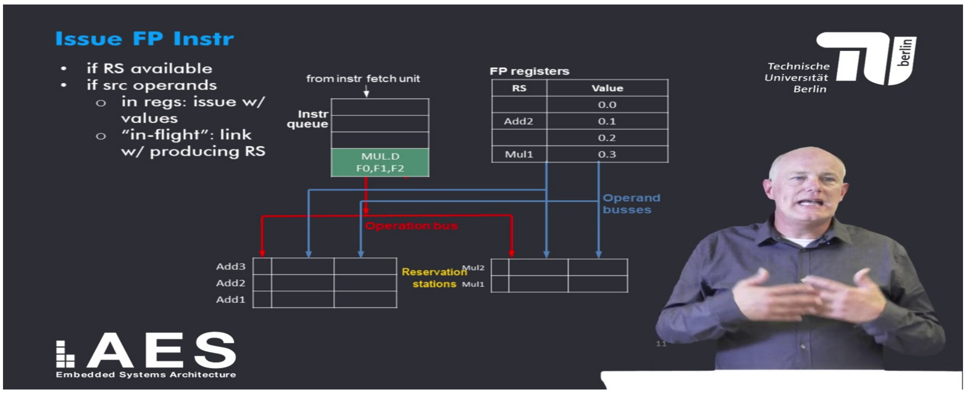
Then the instruction is issued to the reservation station together with these values.The third phase in Tomasulo algorithm is the execute phase when all operands are available, meaning there is no raw hazard and a functional unit that can execute the instruction is available.



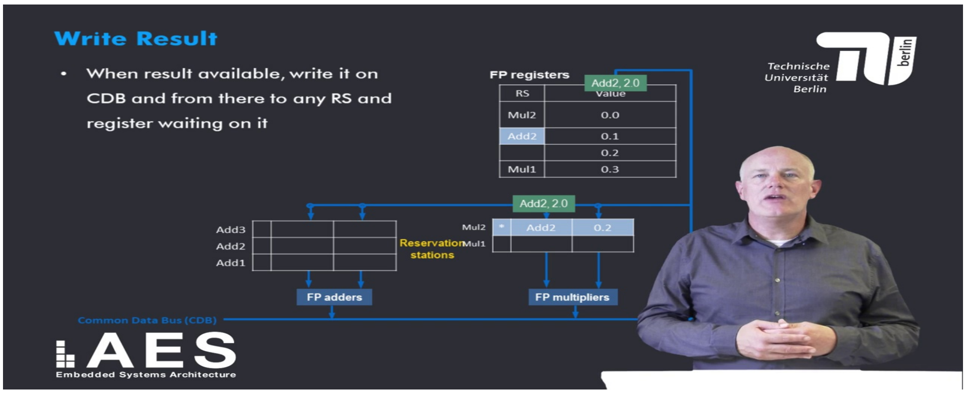
The 4th and final phase is the right result phase.Now I will draw the basic structure of a floating point unit based on Tomasulo algorithm as explained on the previous slide.Then in the second phase, the instructions at the head of the queue is issued to a matching reservation stations. Here I have drawn in total 5 reservation stations, 3 floating point addition register reservation stations that can hold floating point add and subtract operations, and three floating point multiply reservation stations that can hold multiply instructions. The execute phase the instruction is executed. No one.Finally, in the right result phase, the result is broadcast on the common data bus and written to every waiting reservation station as well as the register file.Similarly, the RS2 field contains the identifier of the Rs that will produce a second operant value.



The value of the 1st and of the second operand.55.To track whether an operand value is available in a register or is currently being produced, each register also has an additional field Rs. This field is blank or zero if no currently active instruction computes a result destined for this register.



The multiply instruction will issue 2 reservation station mode 2 for example, highlighted in green. The first operand will be produced by reservation station.2.Finally, execution proceeds to the last stage right result when the functional unit has produced a result.This picture illustrates this face.



0.0 and it's ours field has been cleared and we end up with this state. Thanks for watching in the next lesson. Hoping that you will fully understand it and never forget about it.