

(Refer Slide Time: 21:48)

Instruction Execution (Indirect Mode)

"Load $R_1, (M)$ " (where, M is a memory location)
Task of the instruction: Load the content of Memory Location $M1$ to Register R_1 , where address of $M1$ is specified in memory location M . We assume that length of instruction is 1 (constant).


The three control steps used to fetch the instruction.

1. $PC_{out}, MAR_{in}, Read, Select=0, Add, Zin$
2. $Zout, PCin, WMFC$
3. MDR_{out}, IR_{in}

4. $IR_{out}, MAR_{in}, Read$

In the fourth control step the value of M (i.e., the memory location from which the address of the memory location where data is present) is loaded into the MAR from the IR and the control signals are IR_{out} and MAR_{in} . At the same control step we need to make the memory control signal as READ because the contents of the memory location specified in the IR needs to be loaded into the MDR (in 5th control step after WMFC).

It may be noted that the IR has the entire instruction i.e., Opcode-code for R_1 -address of M . The instruction decoder understands that in the current mode of instruction (indirect) the indirect address of the operand is in the instruction itself and only "address of M " is loaded from IR to MAR



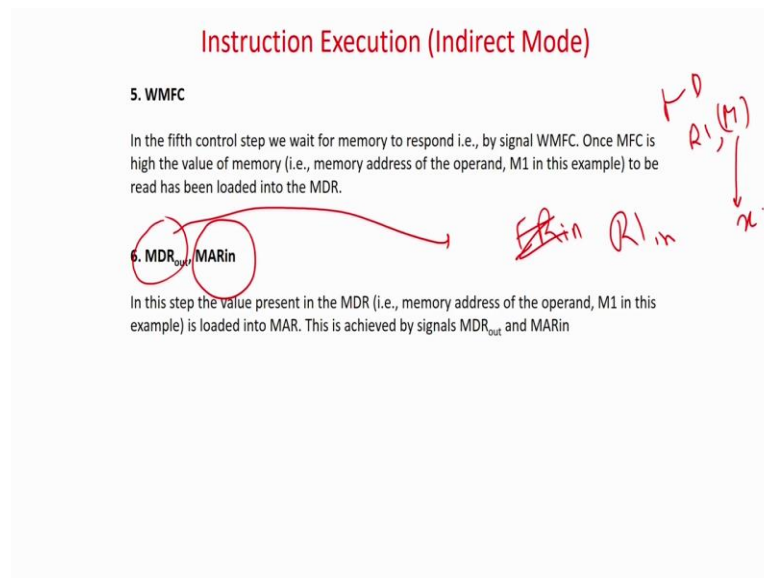
Now, we will go to the another mode, which is more complex in nature which is the indirect mode. By indirect mode already we mean that whenever it's the indirect mode of M ; that means, we say this is the memory here is M , at the address of M here there may be some addresses called x is some content over here, then again you have to look at the content in x and basically this is your operand, this is what is the idea we all know about it. So, if you now look at basically your first 3 stages. So, first 3 stages; as I already were discussing like PC_{out} MAR_{in} and this is control stage basically are only for fetching the instruction.

Next one is what I have to do. So, this is the instruction load R_1 into memory from indirect memory location, that is the content of M you have to again go to that memory location and there will get the operand it has to be loaded to R_1 that is we say that load the control memory location $M1$ to register R_1 where $M1$ is specified in the memory location that is indirect and we assume that the length is one.

So, the next stage is IR_{out} memory register in and read if we are fetching the basically. So, what we are doing here. So, now, the after executing the third instruction the third control step register instruction register IR is having the value of $LOAD R_1, M$. So, in the fourth stage what I am going to do I am going to take the value of M and I am going to feed it into the memory address register so that you can read the value of memory location M that is x in this example.

So, IR_{out} that is you are going to take the value of M and dump it into the memory register address register. So, that now I can read the value of x .

(Refer Slide Time: 23:24)

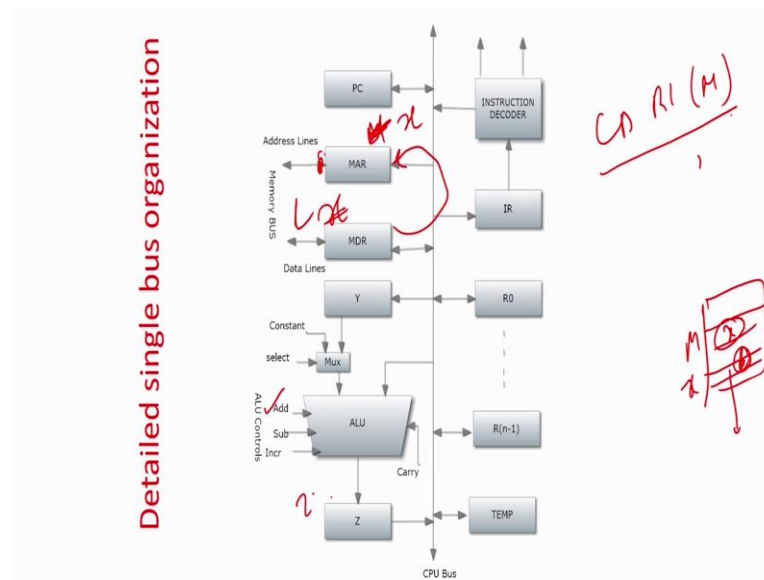


Of course, in stage 5 we have to wait till everything is ready. So, once it is ready basically we know that the value of memory location M that is x in the example is now loaded into the memory data register now interestingly for all cases what we have seen for most of the cases after the memory data out, if it's a direct instruction we generally take this memory data out from the memory buffer register we actually say IR_{in} , if is a fetch or sometimes we call R_1in , in that way.

That means, the operand that is present over here the x we are directly loading at some place or we are directly using it as an instruction, but in this case it is very interesting that the memory data out will be fed to the memory address register in basically; that means, what I am going to take this value of x and I am again dumping it to the memory address register because exact value will be found out in memory location x .

So, if you quickly look at what it happens in the bus it will be more clear. So, look at the bus. So, many times you have to refer to this diagram. So, we will recollect the stage we are in. So, basically we are in this stage when say load R_1 .

(Refer Slide Time: 24:40)



So, we just recall the instruction is something like say load R_1 indirect M and your M we are having something like this M this is x , this is your memory location x and this is the x location. So, now, your instruction register basically has $LOAD R_1, M$ that is already there now what we are doing. So, now, we are making IR_{out} and we are loading the memory address register with M . So, this is done.

So, now, it is done means, now the content of memory location that is x will be given into the memory data register after you have to wait for MFC, now what happened now this content x generally is used by some of the registers if it is the data or it is an instruction it will go to IR , but in this case interestingly what is going to happen this x will be again going to fed back to the memory address register. So, in this case now it will be an x .

So, if it is an x , the content we can say that now the content is 1 . So, that content will again come here and which is actually your real operand. So, in next case what we do we say, memory data register out and memory address register in. So, that the value of x will be fed to the memory address register now the memory buffer register after a wait will give the exact value, which is present in memory location which is your exact data. So, it will, will come to the memory data register and then you can load it to your R_1 or wherever by a simple signal sequence that is MDR_{out} register R_{1in} .

So, that is what is happening right. So, in this case you read the instruction then you read the wait for some time, the signal that is add. Sorry, $LOAD R_1, M$, this instruction is already loaded

into the memory then you take in the sequence you memory data register out and basically you are going it in memory in, that is a very important stage over here that is what going to take the value of content of M , which was x and you are dumping it into the memory address register, next you wait for some amount of time and then in this case now exactly you are going to have the and operand which is present in the memory location x that the content we are assuming l for the time being will be taken from MDR it will be dump to register R_1 . So, in two indirection stage R_1 , we have got the exact operand.

(Refer Slide Time: 26:43)

Instruction Execution (Indirect Mode)

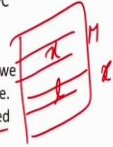
7. WMFC

In the seventh control step we wait for memory to respond i.e., by signal WMFC. Once MFC is high the value of memory (i.e., operand) to be read has been loaded into the MDR.

It may be noted that control signals in 5th and 7th steps are same. However, in the 5th step we read the address of the memory location where operand is stored (i.e., $M1$) in the example. In the 7th control step we read the operand from memory location (whose value is obtained in the 5th step).

8. $MDR_{out}, R1_{in}$

In this step the value present in MDR (i.e., the operand) is loaded into $R1$. This is achieved by signals $MDR_{out}, R1_{in}$.



That was basically first it contains M which was x now this is x and the exact content l which will be actually dumped into R_1 .

So, in the indirect mode we can get the values. So, now, again another mode we are taking which is called registering indirect in this case it was a memory indirect it is a register indirect that you have to go to the register and the content of the register will also contain the location of a memory where the data will be present like.

(Refer Slide Time: 27:09)

Instruction Execution (Reg. Indirect Mode)

"Load R1, (R2)"

Task of the instruction: Load the content of Memory Location M to Register R_1 , where address of M is specified in register R_2 . We assume that length of instruction is 1 (constant).


The three control steps used to fetch the instruction.

1. PC_{out} , MAR_{in} , Read, Select=0, Add, Zin
2. Z_{out} , PC_{in} , WMFC
3. MDR_{out} , Ir_{in}

4. $R2_{out}$, MAR_{in} , Read

In the fourth control step the value of M (i.e., the memory location where data is present) is loaded into the MAR from R_2 and the control signals are $R2_{out}$ and MAR_{in} . It may be noted that R_2 (the register specified as the second operand in the instruction) is specific to the example instruction used in this answer and is not generic. So the control signals to be generated by the instruction decoder is (Register specified as the second operand) $_{out}$.

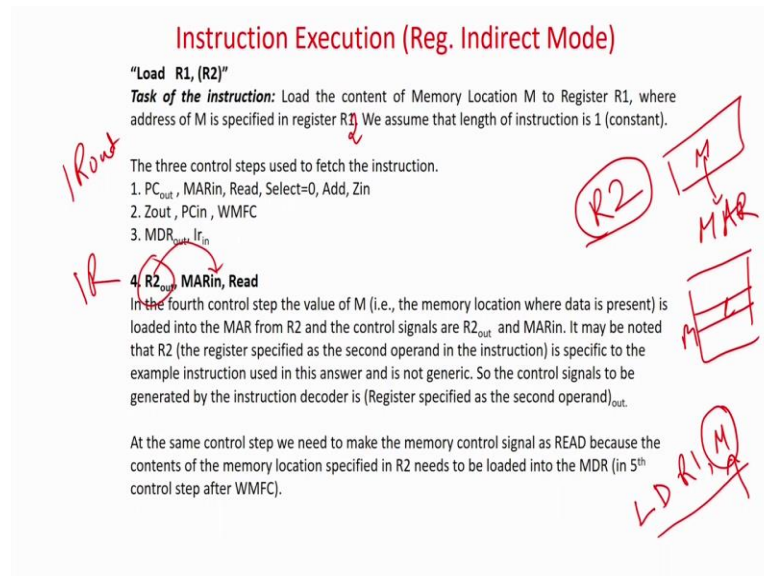
At the same control step we need to make the memory control signal as READ because the contents of the memory location specified in R_2 needs to be loaded into the MDR (in 5th control step after WMFC).



For example, load the content of memory location M to register R_1 , where M is specified in register R_1 that is say there is a register load the content of memory location M to R_1 . So, it should be R_2 . So, load the content of memory location M to register R_1 where M is specified in R_2 . So, R_2 will have some value M . So, it is your memory. So, this is your value of M . So, this is your R_2 .

So, exact value of the operand address will be found in R_2 . So, R_2 will have the value of M , M we will have to be fetched into the looked into the memory this exact value say l will be dumped to R_1 that is what is the register indirect mode that we are going to now do. So, as already discussed many times that the first 3 stages are basically for your basic operation of loading the instruction fetching the instruction then what we do then we say R_2 out and memory address register in. So, now, in this case it is interesting. So, what happened the if you look at R_2 so, R_2 basically is a register where the exact address of the data is present. So, in this case if you see so, what I am trying to do so, if you look at so, your instruction R_2 is register.

(Refer Slide Time: 28:15)



Basically it is your register R_2 , it has the value of M . So, that is your actually memory location address. So, this one you have to feed it directly to the memory address register. So, once you give the value of M , which is the content of R_2 to the memory address register, it will get the value 1 after reading. So, you do register R_2 from memory address M from R_2 actually you are going to dump the value of the memory address M because it will have the value of 1.

So, you note in this case we are not actually reading the value to memory address register from the instruction register for all other cases, if you look so, if you had some instruction like as I told you like say load R_1 say M . So, what we used to do we used load the value of M , from the instruction register directly to memory address register, but in register indirect mode if you observe what we are doing we are directly taking the value of R_2 and we are loading it in the memory address register. So, actually the instruction register when it decodes, it finds that it is an indirect addressing mode.

So, in that case the role of the instruction register is not used to directly hand over all the stuffs to register R_2 and not get involved in the picture, but if it's a direct mode or an indirect mode not involving a register. So, in this case the instruction register value the M has to be directly basically loaded into the memory address register. So, in this case the instruction, the decoder will generate such a signal. So, that instead of R_2 out it will be instruction register out that is it will take the value of M and dump it in your register, here it is a memory direct or memory indirect, but even when the instruction register finds that basically now, the instruction is

involved with register only and no memories in picture. So, directly R_2 the content of R_2 will be directly loaded to the memory address register; that means, basically R_2 , the content M will load to the memory address register after that it is very simple you have to wait for some amount of time and in that case basically. So, once it is done the memory data you have already given it the address of M to the memory address register you have to wait for some amount of time and then this 1 will come to the memory data register or the memory buffer register and it is done then just you have to load it into R_2 . So, you have to wait for some amount of time.

(Refer Slide Time: 30:33)

Instruction Execution (~~Direct Mode~~)
Reg. Indirect

5. WMFC

In the fifth control step we wait for memory to respond i.e., by signal WMFC. Once MFC is high the value of memory (i.e., memory address of the operand specified in R_2 , in this example) to be read has been loaded into the MDR.

6. MDR_{out}, R1_{in}

In this step the value present in MDR (i.e., the operand) is loaded into R_1 . This is achieved by signals MDR_{out}, R1_{in}.

After the memory register you are going to dump it to R in. So, you can see, it is actually in this case it should be I am just a mistake it should be basically register indirect mode register indirect mode just a copy problem. So, in this case if you can see. So, your indirect mode if it's the memory indirect mode you require 8 stages to do it, but if in your register indirect basically, you can solve it in 6, basically 6 steps of time. So, basically register indirect is a more faster mode compared to a or less number of steps required to a memory access, because in this case you just require one access, but in the previous case you required two memory access and in fact, also it depends on whether you want to load or you want to add and all those problems are there.

So, depending on different type of addressing modes the sequence of control or the number of steps will all differ. So, this unit basically gave you a spectrum of different type of addressing modes and with different functionality like load and add and we have shown what are the

sequence of control instructions required or the micro instruction and how a complete instruction can be controlled with this respect to the control signals that is basically which we have covered.

So, we have taken immediate mode, direct mode, indirect mode, register indirect mode, register mode and so forth. So, which gave you a wide spectrum of designing instruction set in terms of basically your control signals given a single bus architecture as well as different type of addressing modes under consideration. So, basically I told you this is a quite small module. So, we come to that and then we have a very simple question like draw the diagram of a CPU with single bus architecture and write different type of instructions with immediate addressing, indirect addressing, displacement addressing.

(Refer Slide Time: 32:05)

Questions and Objectives

Q1: Draw the diagram of a CPU with single bus organization. Write the control steps for completely executing an instruction with

1. Immediate addressing mode.
2. Indirect addressing mode
3. Displacement addressing mode

Briefly explain the actions in terms of control signals carried in each step.

- **Comprehension: Explain:--** Explain the addressing mode with respect to internal structure of the processor and instruction format.
- **Synthesis: Design:--** Explain the Design of complete control steps to execute an instruction that involves different addressing modes, such as, Memory Direct, Memory Indirect, Register Direct, Register Indirect, Immediate, etc.

and some other addressing modes and try to design this try to express how the instructions will execute in terms of control steps and what are the control signals for that, if you were able to answer this of course, the two objectives of this course like explain the addressing mode with respect to internal structure of the processor and instruction format and the control instructions and design the complete control steps to execute instruction that involve different addressing modes gets satisfied.

So, once you, after doing this unit you will be able to solve this problem and you will be able to meet these objectives. So, with this basically we come to an end of this unit and in the next

unit what we will be looking at, we will look at more sophisticated type of instruction like a jump instruction, function call etcetera. So, which will be little bit slight more intricate details.

Thank you.