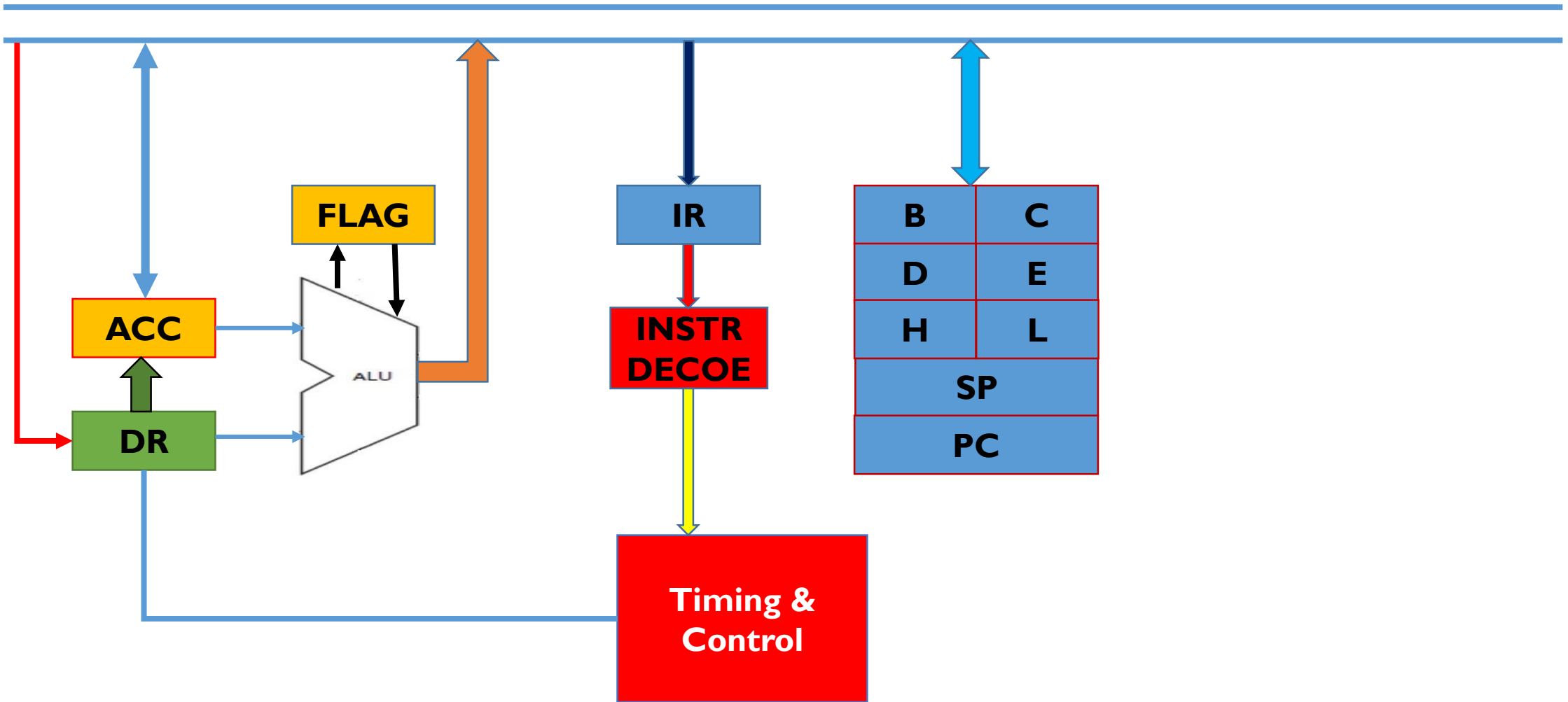
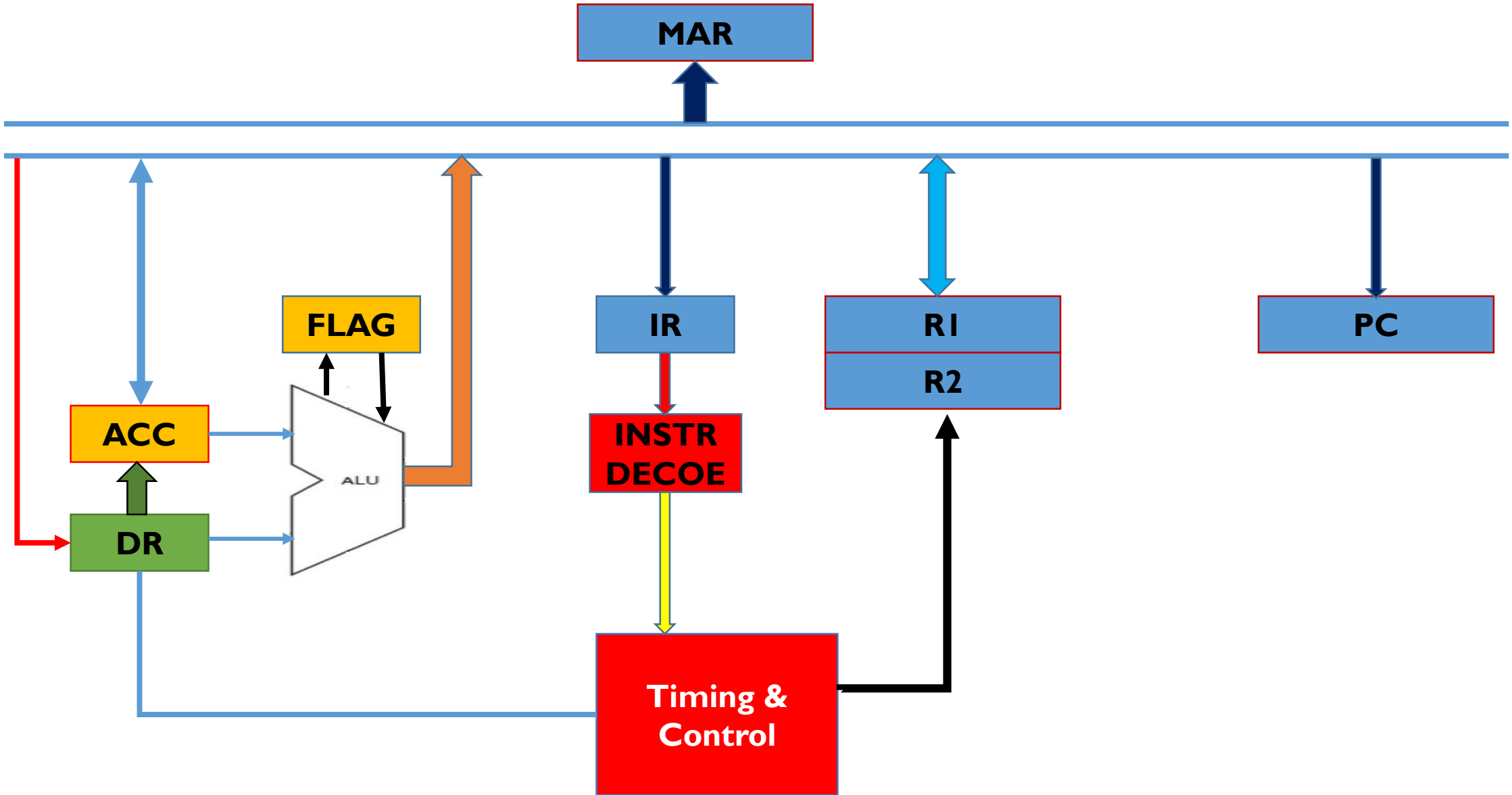
CPU Design - I

- we will take a simplified version of the CPU and try to design that particular CPU.
- So we will not take this entire 8085 but rather a subset of this, just to give you a feel about how the CPU can be designed

8085 ARCHITECTURE



CPU DESIGN



Computer Organization

- ✓ we will have an ALU, it will have an accumulator and it will have a temporary register that is Data Register.
- ✓ let us have two general purpose registers. One is let us say register R1 and other one is register R2.
- ✓ One is let us say register R1 and other one is register R2.
- ✓ we have an instruction register and an instruction decoder.
- ✓ a timing and control circuit and let us have a memory address register.

Computer Organization

✓in case of 8085 where you have a general purpose register that is HL register pair which again acts as the memory address register but in our design we will consider that we have a separate special purpose register, which is memory address register.

- whenever some value is loaded into the memory address register that activates the external address bus, on the external address bus that value is available.

Control Signal

- To see the control signal working let take a simple example.

MOV R1, R2

This assembly code will transfer the data from R2 to R1.

- **What are the control signals that are required?**

Control Signal

- To put the data in R1 we need control signal load.

- Whenever you send a pulse to this load R1 input, load input of register R1, whatever is available at the input of register R1 at that instant that will be loaded into register R1.
- Now this data has to come from register R2.
- whenever I have to transfer a data from a source to some destination, in that case the content of the source must be available on the common data path.

Control Signals

- whenever I have to transfer a data from a source to some destination, in that case the content of the source must be available on the common data path.
- I have more than one registers connected over the common data path and any of the register is capable of sending data onto the common data path.
- R2 must have some control input which when activated will enable R2 to send the data onto the data path. When it is inactive, R2 will not send the data onto the common data path.
- So that particular control signal of register R2 we will mention as output enable of R2.

Computer Organization

MOV R1, R2

Control Signal

MOV R1, R2

LD R1

Control Signal

MOV R1, R2

LD R1

OE R2

Control Signal

MOV R1, R2

LD R1

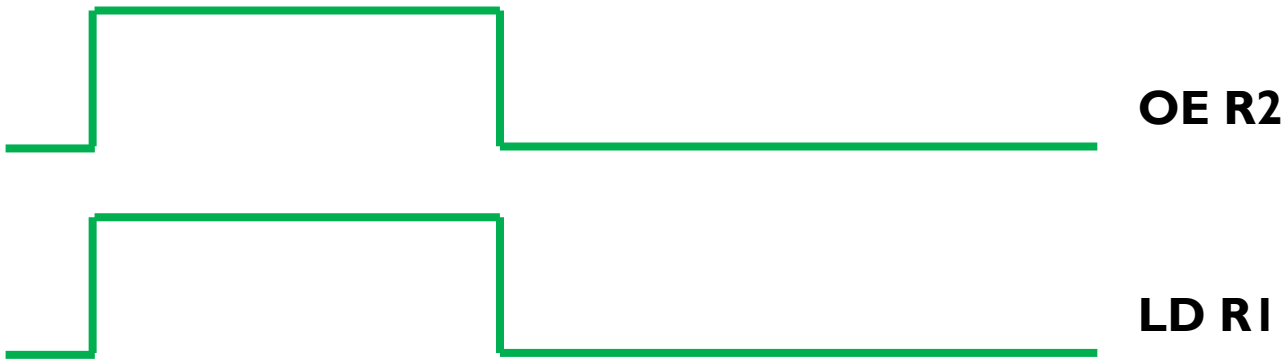
OE R2

Control Signal

MOV R1, R2

LD R1

OE R2

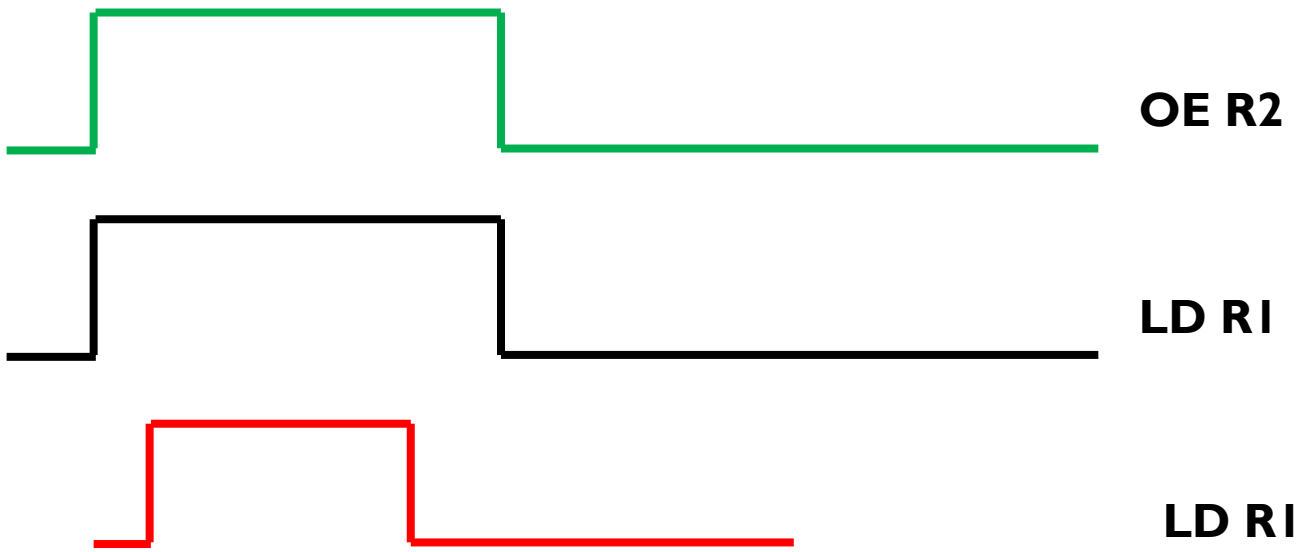


Control Signal

MOV R1, R2

LD R1

OE R2

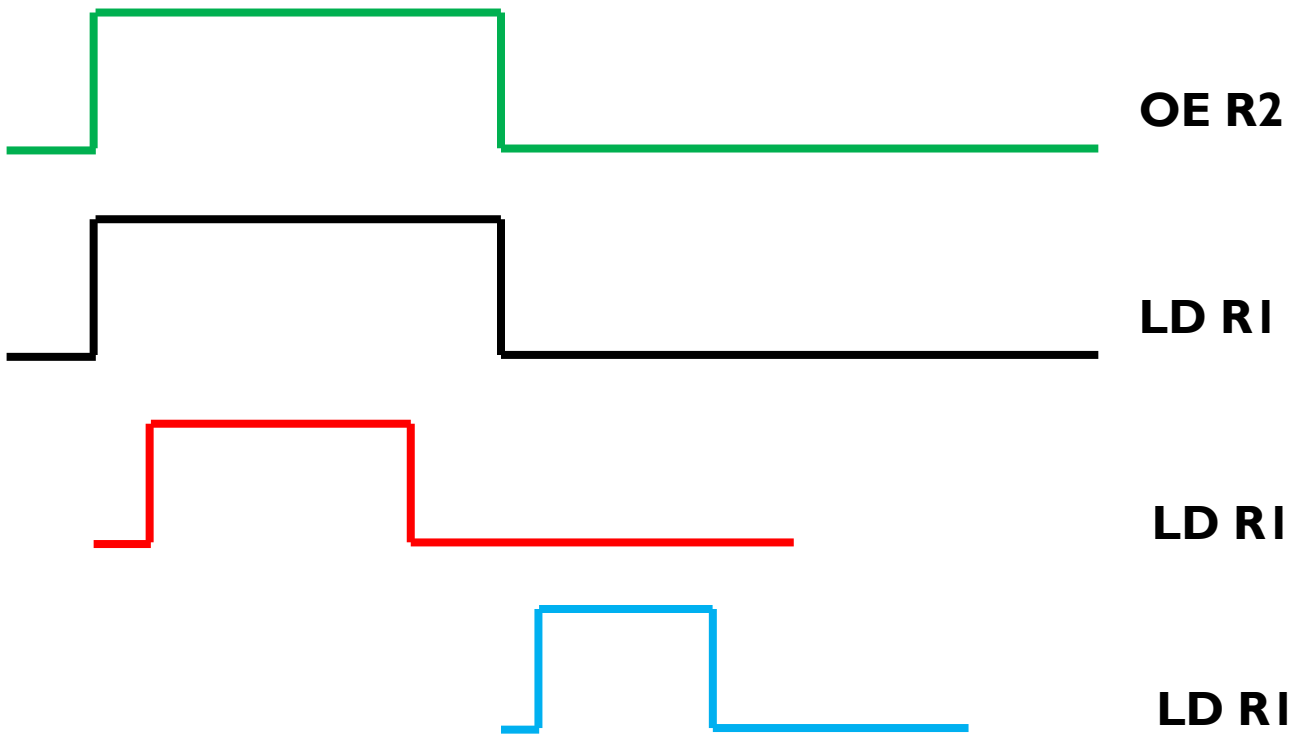


Control Signal

MOV R1, R2

LD R1

OE R2

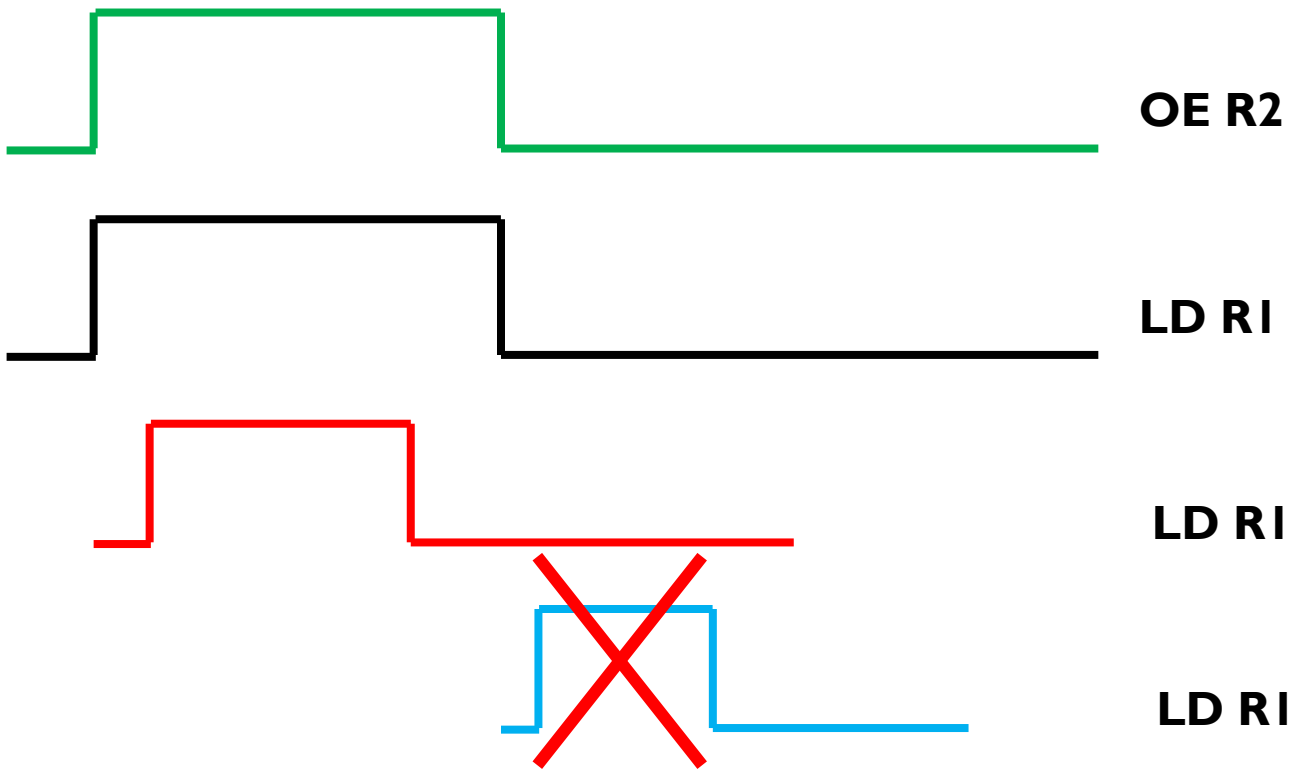


Control Signal

MOV R1, R2

LD R1

OE R2



Computer Organization

- So the situation should be something like this. First you enable the output of R2, so this if I say this is a pulse which enables output of R2. Then load input of R1 should be within this region, it should not go beyond this.
- when the data is actually loaded till that time I must ensure that output enable of R2 is active.

Computer Organization

- ✓for execution of any instruction, you have various stages. for execution of any instruction, firstly you have to read that instruction from
- ✓the main memory, put it into instruction register then the instruction will be decoded and depending upon the decoder output, it's the timing and control unit that has to decide that what are the control signals that will be generated and in which sequence they will be generated.
- ✓after instruction read, the instruction has to be decoded. So you have decode instruction first and after decoding the instruction then only the instruction will be executed.
- ✓the next cycle is execute instruction.

Computer Organization

- ✓At the time of execution the instruction can be of various kinds.
- ✓Above discussed instruction is Register Reference Instruction.
- ✓Other instructions may be reading a data from the memory where the first instruction, read instruction decode will be identical but while executing the instruction, it has to invoke another memory read or memory write operation which is again similar to instruction read operation.

CONTROL SIGNALS

- Let assume :

ADD R1

CONTROL SIGNALS

❖Let assume :

ADD R1

MEANS ACC ← ACC + R1

So, the above instruction is a Register Reference Instruction. Because both operands are registers.

CONTROL SIGNALS

❖Let assume :

ADD R1

MEANS ACC ← ACC + R1

So, the above instruction is a Register Reference Instruction. Because both operands are registers.

Multiplication also possible by ADD operation.

Instructions

- Subtraction is possible by Compliment operation only.

CMP

Instructions

❖Subtraction is possible by Compliment operation only.

CMP

$$\boldsymbol{ACC \leftarrow \overline{ACC}}$$

✓This complement will perform the operation of accumulator gets the complement of the accumulator.

✓the $\bar{Q}$ output to D input and activate the load in that case whatever is $\bar{Q}$ that will be loaded into the flip flop.

✓DIVISION CAN BE PERFORMED BY SUBTRACTION.

LOGICAL OPERATION

AND R1

ACC ← R1 ^ ACC

- ✓This will perform the operation of logically ANDing the content of register R1 with the content of the accumulator and storing the result in the accumulator.
- ✓So we find that when I have this AND and CMP together that means I can perform NAND operation and if I am able to perform NAND operation.
- ✓I am able to perform any type of digital operation, logical operation because NAND function is functionally complete.

JUMP

JMP

PC ← JMPADDR

- ✓ **Let me assume that I have just a jump instruction. What are the other instructions that you need? Again compare can be implemented by subtraction. Yeah, MOV that is called a data transfer instruction. So I have to have MOV instruction**

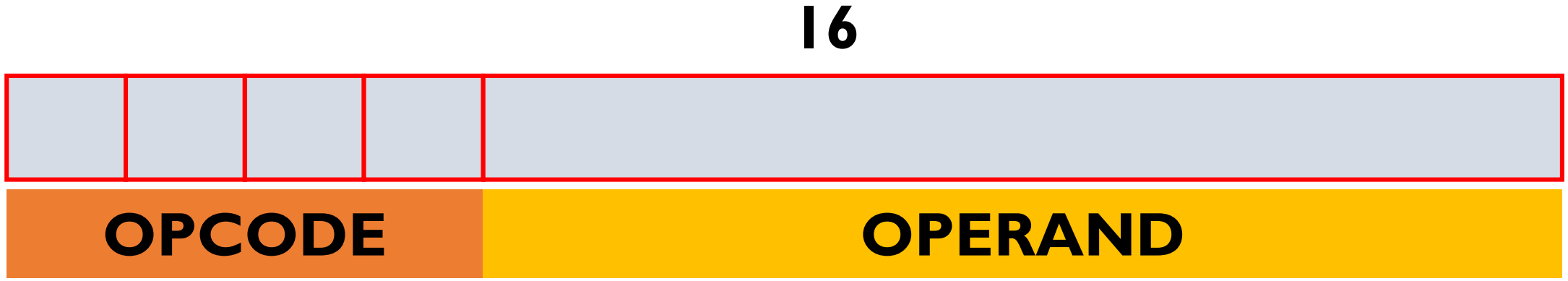
Computer Organization

- Jump instruction we will load, jump address into program counter because you already know that whatever is the content of the program counter, the program execution starts from that address.
- Now once you decide what are the instructions, the next step is you have to decide that what will be the instruction format.
- So instruction format is you have to decide that how many bits or what will be length of every instruction.
- How many bits in that instruction will be allocated for allocated as output which will identify what instruction it is and how many bits will be reserved for operand.

Computer Organization

- You see that in this case we have two types of operand. In some cases, the operands are registers, for some of the instructions the operands can be memory locations.

Computer Organization



- I assume that the instruction length is say 16 bit.
- out of which these 4 bits have been used as instruction opcode.
- the remaining 12 bits can be used as operand.

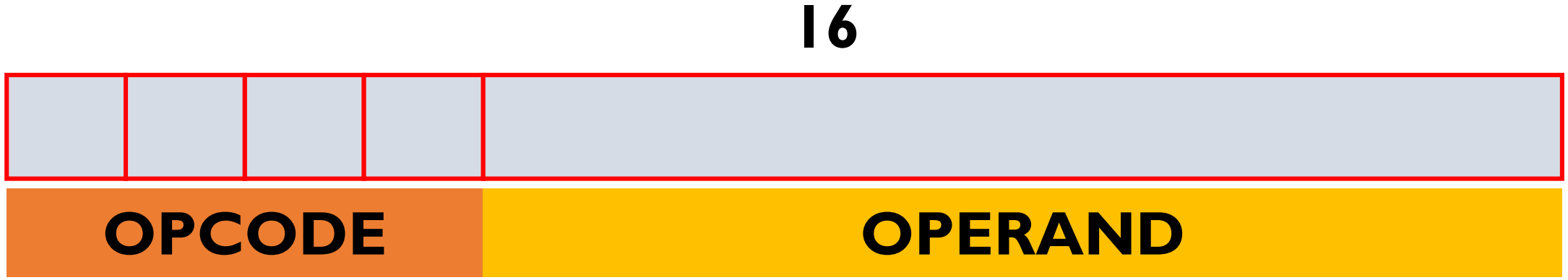
Computer Organization

- though we have 13 different instructions but you find that these instructions can be grouped as register reference instructions and memory reference instructions.

- if OPCODE bit is a 1 1 1, I can say that whenever the opcode field is 1 1 1, the instruction is a register reference instruction.

- For register reference instruction I need not give any operand address because the operands are within the CPU registers themselves and there we have very, very limited instructions

Computer Organization



- 1 1 1 → Register Instruction
- 0 0 1 → MOV ACC, M
- 0 1 1 → JMP
- 0 1 0 → MOV M, ACC

Computer Organization

- out of 13 I have two instructions which are actually memory reference instructions and the remaining 11 are register reference instructions.

- For operand I have bits from B0 to B11.

- For memory reference these bits will give address of memory and for Register it will identify a register uniquely.

Computer Organization

B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 B11

1 0 0 0 0 0 0 0 0 0 0 0

THEN IT IS → ADD R1

AT THIS MOMENT OPCODE WILL BE 1 1 1

0 1 0 0 0 0 0 0 0 0 0 0

IT IS **CMP**

001000000000

IT IS AND R1

Computer Organization

- let us see that given this instruction, set of instructions and the instruction format, how we can design the hardware?
- for designing the hardware we have to study each of these instruction in details that what these instructions are doing and in which sequence those operations will be done.

Computer Organization

- whenever the CPU will execute any instruction, the first one as we have said is an **opcode fetch cycle.**

- the instruction has to be read from the main memory and it has to be put into the instruction register.

- After that it will be decoded, the signal will be given to the control and timing circuit.

Computer Organization

ADD R1	ACC ← ACC + R1
CMP	ACC ← $\overline{ACC}$
AND R1	ACC ← R1^ACC
JMP	PC ← JMPADDR
MOV R1, R2	
MOV R2, R1	
MOV ACC, R1	
MOV ACC, R2	
MOV R1, ACC	
MOV R2, ACC	
MOV ACC, M	
MOV M, ACC	
MVI	

Computer Organization

- The control and timing circuit will generate the control signals, the required control signals in the required signals.

- the first cycle that is opcode fetch cycle, it is common for each and every instruction.

- So let us see how this opcode fetch cycle will be performed.

Computer Organization

- For opcode fetch as we have said that this is similar to a memory read operation.
- the memory address register has to be set with the address of the instruction that is going to be fetched.
- we know that the address of the instruction which will be executed resides in the program counter.
- the first operation is setting the data into the memory address register or address into the memory address register and this will come from the program counter.

Computer Organization

- So once the content of program counter goes to the memory address register then that particular address, that particular location in the memory has to be read and whatever you read from that address has to be put into the instruction register.

- NEXT OPERATION is instruction register gets the value from memory whose address comes from memory address register.

- Because once you read an instruction, the program counter has to be incremented so that it points to the next instruction in the memory.

Computer Organization

- So I simply increment the program counter by one, assuming that each instruction is using one memory location only.

- Now after that you have to decode the instruction. So decode the content of instruction register.

So these are the operations which are to be performed always.

Computer Organization

T0 : MAR ← PC

T1 : IR ← M[MAR]; PC ← PC + 1

T2 : DCD(IR)

Computer Organization

- So I simply increment the program counter by one, assuming that each instruction is using one memory location only.

- Now after that you have to decode the instruction. So decode the content of instruction register.

- I can send the content from lower 12 bits of the instruction register to memory address register with decoding instruction.

So these are the operations which are to be performed always.

Computer Organization

T0 : MAR ← PC

T1 : IR ← M[MAR]; PC ← PC + 1

T2 : DCD(IR) ; MAR ← IR0-11

- So what I will do is I will shift the lower 12 bits of the instruction register that is IR0 to IR11 to the memory address register during the same time when the instruction is decoded.

- These are the operations which are common for execution of any instruction and the operations are to be done in this sequence.

Computer Organization

- So I will define this timing intervals as T0, T1 and T2. I am referring T as similar in 8085 microprocessor.

- These timing intervals are popularly known as machine states.

- So these are different machine states T0, T1 and T2. During T0 the operation is specific, during T1 the operation is specific, during T2 the operation is also specific. T3 onwards the operations will be dictated by the decoder output.

Computer Organization

- So let us take few of the instructions. See what is to be done during T3 T4….
- Let me consider the first instruction say ADD R1.
- the addition operation has to be performed by this ALU which needs two operands.
- one of the operands is in the accumulator, the other operand is in register R1 but R1 is not directly connected to accumulator.

Computer Organization

- So the ALU is getting data from accumulator and the data register DR.

- One of the operands is in the accumulator, the other operand we have to load into the data register.

- So that means from R1 that data has to be transferred to date register before this addition operation can be performed. So that is the first step.

T3 : DR ← R1

Computer Organization

- DR and ACC both are directly connected to ALU, so no extra signal is required to activate ALU.

T4 : ALUadd(ACC, DR)

Computer Organization

- These type of operations are called **micro-operations.**

- **Here I am considering that ALU will also complete its operation in one time UNIT. Sometime it will take more time than normal. Then Time cycles will increase.**

Computer Organization

MOV R1, R2

Computer Organization

MOV R1, R2

Both the Registers are connected and can transfer the data only in one cycle.

T3 : R1 ← R2, T0

And again set the machine to T0.

Computer Organization

MOV ACC, M

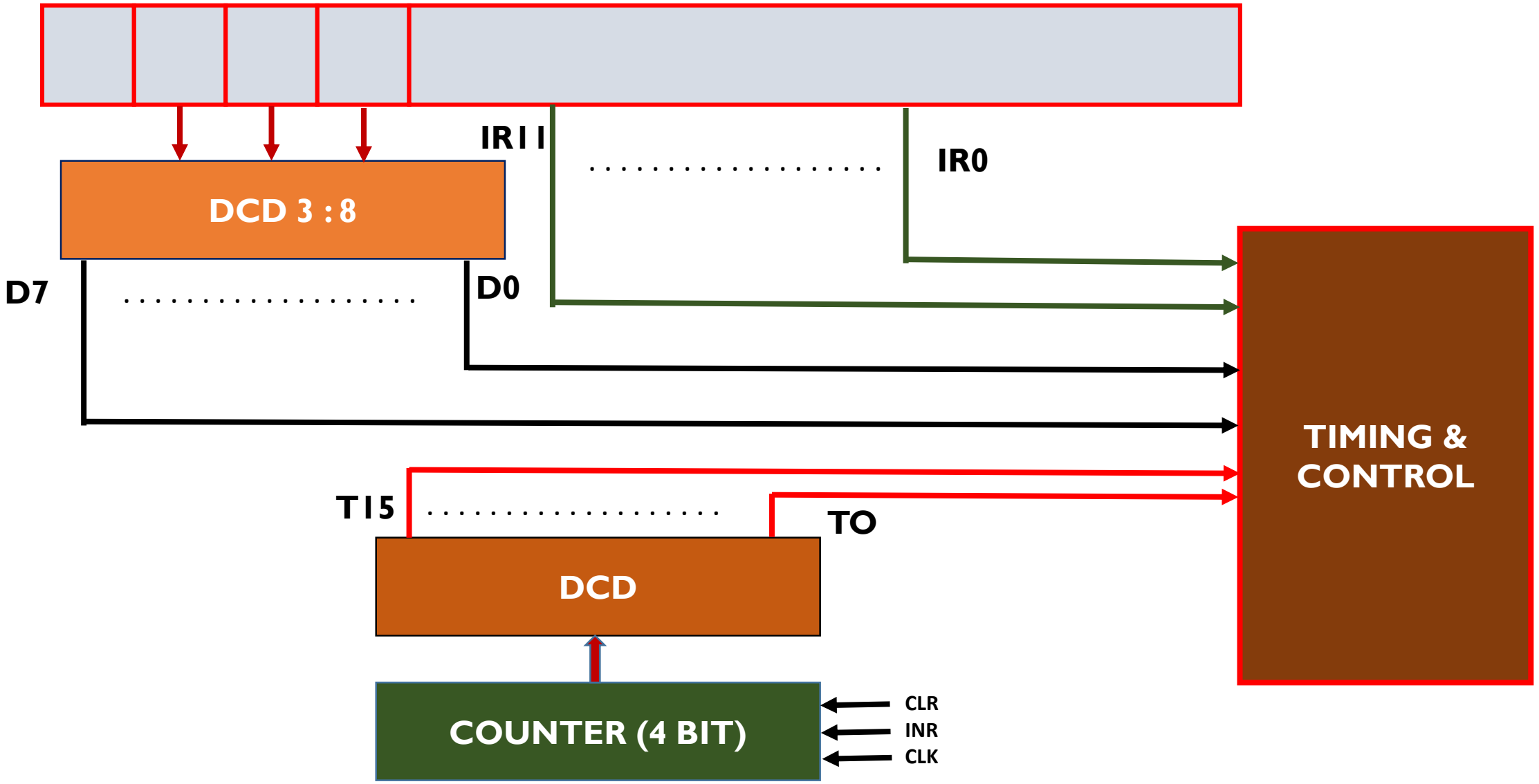
Computer Organization

MOV ACC, M

T3 : ACC ← M[MAR], T0

Because at T2 the memory address is already transferred to MAR. So, directly we can transfer the data from Memory to ACC.

Hardware Implementation



Hardware Implementation

- I need two specific units.

- One for decoding the instruction, the other unit for generating the machine states or generating the time intervals T0, T1, T2, T3 and so on.

- Now what is this simplest way of generating the machine states? You simply use a counter, output of the counter you fit to a decoder.

Hardware Implementation

- The counter output will be fed to a decoder.
- One of the decoder outputs will be active.
- So here when I assume that my counter is a 4-bit counter that means I can have 16 different machine states that is T0 to T15 and none of the instructions.
- I need to have is an instruction decoder.
- Out of this right now we are not making use of the most significant bit that is IR15.

Hardware Implementation

- The next three bits I am saying that this contains the opcode of the instruction.
- I have a decoder which is a 3 to 8 decoder because it takes 3 inputs from the instruction register and it generates 8 outputs D0 to D7.
- Counter will have one input of clear, that is the set, counter will have one input called increment, counter will have another input called clock.

Hardware Implementation

- This clock is the master clock or assumption is when this increment input is one, with every clock pulse the counter will be incremented by one.

- When the clear input is one, with this clock pulse the counter will be clear to 0 and that is what will enable us to set the machine state to zero after performing desired operation.

Hardware Implementation

- You all know that whenever you switch on the power of 8085 or you reset a 8085, the program counter of 8085 becomes 0 and you all know that the zeroth location in the main memory must contain an instruction.
- That means initially the sequence counter will be set to state zero and during time state T0, the operation is specific. This is hardware specific, this is not programmable.

Hardware Implementation

- So if at, in the zeroth location in the main memory for an 8085 based system I put anything other than an instruction, the instruction decoder will not give me any value. It will give me some garbage and timing and control circuit cannot recognize that, it cannot generate any proper timing control signal.
- Typically, what you put is a jump instruction and with that jump instruction you come to a bigger routine which is your main program and that location is different for different CPU's.

Working

- during time interval T0, your program counter content has to be loaded into memory address register.
- I have to activate the output enable of program counter.

$$PC : OE = T0+$$

- I have to activate the load input of memory address register.

$$PC : LD = T0 + T2$$

Working

- During T I what are the registers that are involved? one is instruction register otherwise the program counter. So I have the next register which is instruction register.
- The control signals that are encountered till now is for instruction register, it is the load input.

IR : LD = T I +

Computer Organization

- For program counter it is the increment input INR.

PC : INR = T1 +

- During time interval T2, you find that the instruction is to be decoded.
- However, during time interval T2 again I am loading the content of instruction register 0 to 11, the lower 12 bits of the instruction register into memory address register.

IR : OE = T2 +

Computer Organization

- the output enable control will decide the output of instruction register going to the common data path and this must be active during time interval T2.

- During time interval T2, the content of instruction register goes to memory address register.

- That means the load input of memory address register also must be active during time interval T2.

Computer Organization

MAR : LD = T0 + T2

- Now T3 onwards, the decoder output will come into picture.
- So for that let us consider this ADD R1 instruction.
- D7 output of the decoder will be high because we have said that 1 1 1 in the opcode tells you that it is a register instruction

Computer Organization

- During time interval T3, if I find that D7 is high.
- at the same time it is ADD operation if I0, the zeroth bit in the operand field of the instruction register is high then it is ADD R1.
- it is time interval T3 and D7 is high and I0 is high.

R1 : OE = T3D7I0

Computer Organization

- I have to transfer the data from R1 to data register.
- I have to activate load input of the data register.
- if the machine is in state T3 and instruction decoder output D7 is high and the instruction register bit I0 is high. Whenever this is true, the load input of register, DR data register must be active.

DR : LD = T3D7I0

WORKING

- for performing the addition operation, you find that during time interval T4, accumulator will get the output of the ALU and because this is ADD operation, so ALU has to perform the addition operation.
- ALU will be involved for performing only two operations, one is ADD operation and the another one is AND operation. (All other operations can be form with complement)

Working

- So in the simplest case, we will assume that ALU will also have two mode select inputs.
- One corresponding to ADD, the other one corresponding to AND.
- For performing this ADD operation coming to the control signal needed for the accumulator, what we need is accumulator has to have a load input.

Working

- So for the accumulator will have a load input because the result after addition will be loaded into the accumulator.
- It is to be performed during the time state T4.
- So this load input of the accumulator will be active during T4 when D7 is active because this is register reference operation and I0 is high.

ALU : ADD = T3D7I0

Computer Organization

- the other unit that is involved in the timing and control is sequence counter.
- When this increment output will be active, it has to be active during T0 because after T0 the machine state has to go to T1. It also has to be active during T1 because after T1, the machine will go to T2. It also has to be active during T2 because after T2, the machine has to go to T3. It also has to be active during T3 because after T3 it will go to T4.

SEQ CNT : INR = T0 + T1 + T2 + T3D7I0 +

Computer Organization

- So after completion of the execution of any instruction, the machine has to go back to time state T0.
- at the end of machine state T4, we have to bring back the machine state to T0.
- So this clear input has to be active, if during T4 if D7 is high and I0 is also high.

CLR = T4D7I0 +

MOV R1, R2

- during time state T3 and the control signals that is required for performing this operation is output enable of R2 and load input of R1.

- All the operations had been performed within the register and the I3 bit of the instruction register has to be high.

- That means for this instruction, we have to have the opcode as 1, 1, 1 that means D7 output will be high and the bit I3 will also be high.

R2 : OE = T3D7I3 +

MOV R1, R2

- During the same interval, the load input of register R1 also has to be active because the data has to go from register R2 to register R1.

R1 : LD = T3D7I3 +

- if we set this conditions true then the data will be transferred from register R2 to register R1.
- after T3, the sequence counter has to generate machine state T0.

MOV R1, R2

- So, for the clear input, we must have T3, D7 and I3.
- So clear becomes T4 D7 I0 or T3 D7 I3.

CLR = T4D7L0 + T3D7I3 +

MOV ACC, M

- that is reading the data from a location in memory and loading that data into accumulator.

- for that we need memory address from which location in the memory that data has to be read and you see that during time interval T2.

- So we don't have to perform any extra operation for that purpose

MOV ACC, M

- we have to simply generate the memory read control signal and we have to generate the accumulator load control signal during time interval T3.
- So MR will be active, if during time state T3 if D1 is high.

ACC ← M[MAR] :T0

M : MR = T3D1 + T1 +

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization

Computer Organization