# Microprogrammed Control

- we have discussed that is called hardware control unit because all the control signals are generated by hardware circuit.
- there is another way of generating the control signals and that is called a micro programmed control unit.
- it is generated through software and because these control signals are generated through software this is more flexible.

# Microprogrammed Control

**HARDWARE CONTROL BASIC OPERATIONS ARE AS FOLLOWS**

**T0 : MAR ← PC**

**T1 : IR ← M[MAR]; PC ← PC + 1**

**T2 : DCD(IR) ; MAR ← IR0:11**

# Microprogrammed Control

- during time state T0 the operation that was performed was the memory address register, gets the content of the program counter.

- during time interval T1 or machine state T1 what we had done is, instruction register gets the value from memory whose address is in the memory address register.

- the program counter is incremented by one and during T2 the content of the instruction register is decoded.

# Microprogrammed Control

$$LD_{MAR,}\ OE_{PC} \rightarrow T_0$$
$$LD_{IR,}\ RD_{M,}\ PC_{INR,} \rightarrow T_1$$
$$LD_{MAR,}\ OE_{IR} \rightarrow T_2$$

# Microprogrammed Control

- the load control input of the memory address register
- output enable of the program counter.

- So both of these control signals are to be activated during machine state T0.

- to perform a memory read operation, so we have to activate the read control signal which will go to the memory.

# Microprogrammed Control

- The program counter is incremented, so for the program counter we have to activate the increment control signal.
- These are the control signals which have to be activated during machine state T1.
- Similarly during machine state T2, we said that decoder is a combinational logic circuit. So for that we don't need any control signal to be activated.

# Microprogrammed Control

- But for the memory address register, when it is getting loaded from the instruction register, we have to again activate the load control input of the memory address register and we also have to activate the output enable control signal of the instruction register. So these are the control signals that have to be activated during time state T2.

- So in case of hardware control circuit, what we have done is we have ensured generation of these control signals in this proper sequence by making use of the instruction decoder and the sequence counter.

# Microprogrammed Control

- Now there is another way of generating these control signals.
- identify total number of control signals that are required.
- if number of control signals is equal to say **n** then we can generate these control signals in the given sequence.

# Microprogrammed Control

**No . of control signals = n**

$$C_i \rightarrow i^{th}\ bit\ in\ a\ memory\ location$$

- ✓**So a control signal Ci will be represented by the ith bit in a memory location.**

# Microprogrammed Control

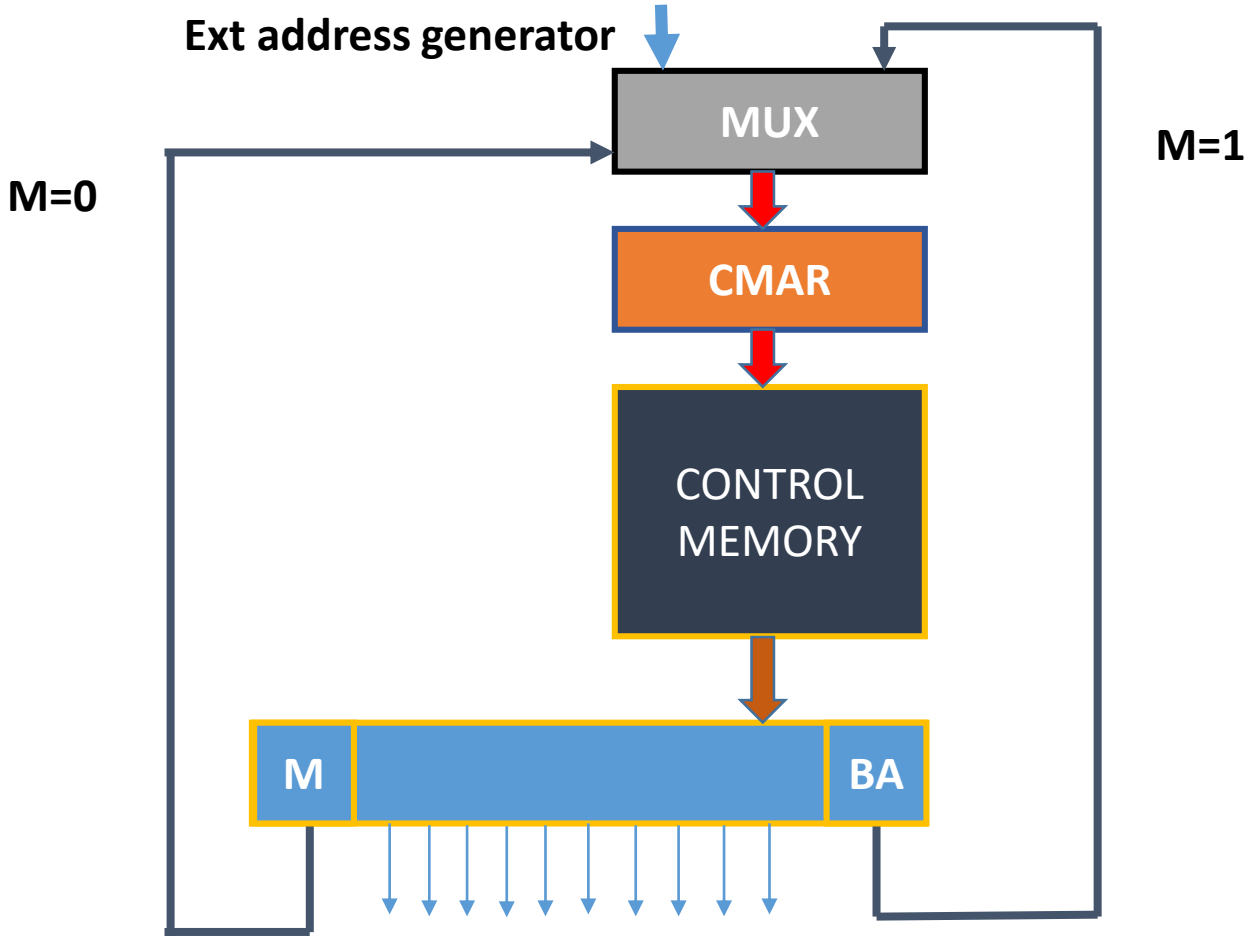
- when ith bit is equal to zero in a particular memory location and when that memory location is read, the control signal Ci will not be activated.

- if **ith** bit equal to one and I read that particular location in the memory then control signal **Ci** will be generated.

# Microprogrammed Control

- I can associate every control signal with a corresponding bit in a memory location and read the memory locations in the desired sequence.

# Architecture MPC




micro program sequencer

# Microprogrammed Control

- for storing these control signals I need a memory.
- in this case the memory is called a control memory because it is storing the control signals.
- I must have a register which tells you what is the address in the control memory that is to be read.
- we have a control memory address register or CMAR.

# Microprogrammed Control

- Output of that particular location, I can load into another register.
- A part of this will give you the control signals.
- an external resource generator or address generator which after decoding the instruction, identifies the location in the control memory from where the required control signals will be obtained.

# Microprogrammed Control

- One of the inputs to the multiplexer is the branch address which comes from the control memory location itself. The other input to the multiplexer comes from the address generator or external address generator.

- M = 0, the address will be taken from the external address generator.

- If it is equal to 1 then address will be taken from the branch address field of the control memory location.

# Microprogrammed Control

- So this will be the overall architecture of a micro programmed control unit where the control signals are encoded within the control memory itself and this can be a ROM.

- So all the components except the control memory take part in deciding the next location in the control memory that is to be read. So it is called micro program sequencer

# Microprogrammed Control

- Now let us see that how these control signals that is needed for all purpose can be generated through this micro program control unit.

# Microprogrammed Control

$$LD_{MAR} \rightarrow C_0$$

$$OE_{PC} \rightarrow C_1$$

$$LD_{IR} \rightarrow C_2$$

$$RD_{M} \rightarrow C_3$$

$$PC_{INR} \rightarrow C_4$$

$$OE_{IR} \rightarrow C_5$$

# Microprogrammed Control

- let me say that load memory address register load memory address register that is associated with bit number C0 that is the zeroth bit in the control memory.

- Output enable of the program counter that is associated with bit number one in the control memory or C1.

- load instruction register is associated with bit number C2 in the control memory.

# Microprogrammed Control

- read control signal for memory is associated with C3 in the control memory.
- program counter increment that is associated with bit number C4 in the control memory then load memory address register is already considered.
- Then what we need is output enable of instruction register is associated with bit number C5 in the control memory.

# Microprogrammed Control

- Let check the signals for some operations:
- the control signals that are required during the machine states T0, T1 and T2.
- let me put it this way, that I put control signal C0 here, C1 here, C2, C3, C4 and C5.
- during time state T0, the control signals that are required are output enable of program counter and load memory address register.

# Microprogrammed Control

- I will put C0 equal to 1 because that is the control signal that is required during time state T0 and I have to have output enable of program counter because program counter is loaded into control memory address register.

- output enable of program counter which is associated with bit number C1, so I also said C1 equal to 1.

# Microprogrammed Control

- I will put C0 equal to 1 because that is the control signal that is required during time state T0 and I have to have output enable of program counter because program counter is loaded into control memory address register.

- output enable of program counter which is associated with bit number C1, so I also said C1 equal to 1.

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# Microprogrammed Control

- during machine state T1, the control signals that are required is load instruction register which is associated with bit number C2.
- in the next control memory location and I also need read control signal of the memory which is associated with bit number C3.
- at the same time, the program counter is to be incremented by one. So increment of the program counter which is associated with bit number C4.

# Microprogrammed Control

- during machine state T1, the control signals that are required is load instruction register which is associated with bit number C2.
- in the next control memory location and I also need read control signal of the memory which is associated with bit number C3.
- at the same time, the program counter is to be incremented by one. So increment of the program counter which is associated with bit number C4.

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

# Microprogrammed Control

- during time state T2, the control signals that is required are output enable of the instruction register which is associated with bit number C5. So I set C5 is equal to 1
- I also need load of the memory address register to be activated which is associated with bit number C0. So I also set C0 equal to 1.

# Microprogrammed Control

- during time state T2, the control signals that is required are output enable of the instruction register which is associated with bit number C5. So I set C5 is equal to 1
- I also need load of the memory address register to be activated which is associated with bit number C0. So I also set C0 equal to 1.

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

# Microprogrammed Control

- find that initially when you power on the machine, the first operation that is to be performed is an opcode fetch.
- ensure that whenever the machine is powered on or whenever the machine is reset, the control memory address register will also be reset to zero.
- the micro programmed controlled unit will start generating control signals from the zeroth location in the control memory.

# Microprogrammed Control

- That means the first control signals it will generate are C0 control signals associated with C0 and C1 which are nothing but load memory address register and output enable of the program counter. From the program counter the address goes to memory address register

# Microprogrammed Control

- every location in the control memory will have two more fields.
- One is the next address field where from the next control signals are to be read and it also have a field, one bit field which is the address select field.
- find that after generating these control signals, the next control signals are to be generated from the next memory location within the memory which is memory location one.

# Microprogrammed Control

put the next memory location in few more bits. I put it as 0 1.

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 | BA | M |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 01 | 1 |

# Microprogrammed Control

- next time the control signals will be generated are C4, C3 and C2 and C4, C3, C2 means increment program counter, read memory and load instruction register.
- So after these control signals are generated, the next control signals are to be read register from the third location in the control memory.
- again put in the next address field as 0 2, so you will find that I am putting in the decimal point.

# Microprogrammed Control

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 | BA | M |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 02 | 1 |

# Microprogrammed Control

- After that the instructions are to be decoded and depending upon the decoder output, the control signals have to be generated and that is the actual execution phase of the instruction.
- I put in this branch address location that is immaterial because the address now is to be generated by the external source but for that what you have to do is we have to set the mode field accordingly. So as we have said that whenever M is set to 1.

# Microprogrammed Control

- If the M is set to 0 then the next address is taken from the external address generator.

# MOV R1, R2

- Instruction during time state T3,
- OER2 and the other control signal that is to be generated is LDR1.
- C6 is the bit which is associated with output enable of R2 and say C7 is the bit which is associated with LDR1.
- Once this instruction is executed then I have to go back to instruction fetch cycle that means time state T0 and for time state T0 onwards, the opcodes are stored starting from location 0.

# MOV R1, R2

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 | BA | M |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

- after completing execution of this instruction, the control memory address register will be loaded with value 0 0 because the M, I have set to 1 and as the control memory address register is loaded with value 0 0

# MOV R1, R2

- the value of L will depend upon which instruction has been fetched and put into instruction register and the decoder, instruction decoder will decide the value of L.
- So this is how I can write a software so I can put it this way that this is nothing but a software for execution of this instruction itself.

# MOV R1, R2

| | C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 | BA | M |
|---|---|---|---|---|---|---|---|---|---|---|
| L | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

# MOV R1, R2

- the value of L will depend upon which instruction has been fetched and put into instruction register and the decoder, instruction decoder will decide the value of L.

# ADD R1

- At T3 the content of R1 has to be loaded to data register then during machine state T4 accumulator is to be loaded with sum of accumulator and data register.
- we had put in this way, ALU performing add operation on accumulator and data register.

**T3 : DR ←R1**

**T4 :** $ACC <- ALU_{ADD}(ACC, DR)$

# ADD R1

- the control signals that are needed are load data register and output enable of R1.
- during machine state T4, the control signals which are required are load accumulator.
- Then we need ALU add control signal because when this control signal is made equal to 1 then only ALU will perform add operation.

# ADD R1

- Simultaneously we also need output enable of ALU because from the ALU, output of the ALU is going to the common data path.

- So I also have to activate output enable of ALU.

- I put assign load data register to control signal say C8, output enable of R1 to C9, load accumulator to C10.

# ADD RI

- In the control signals that we have considered so far these control signals are not considered. So I have to have new bits for generation of this control signals.

- So for ALU add I need C11 then output enable of ALU is C12.

- So I put it this way, I have to have bits C8, C9, C10, C11, C12.

# ADD RI

- ✓Then during machine state T3, the bits which are needed are C8 and C9.
- ✓I put these two bits equal to one. The remaining bits will be zero.
- ✓During machine state T4, I need the control signals load accumulator, so that is C10 which will be equal to 1.
- ✓need ALU add control signals C11 that will also be equal to one.
- ✓Output enable of ALU that is also equal to one.

# ADD RI

$$LD_{DR}$$

$$OE_{R1}$$

$$LD_{ACC}$$

$$ALU_{ADD}$$

$$OE_{ALU}$$

# ADD RI

$$LD_{DR} \rightarrow C_8$$
$$OE_{R1} \rightarrow C_9$$
$$LD_{ACC} \rightarrow C_{10}$$
$$ALU_{ADD} \rightarrow C_{11}$$
$$OE_{ALU} \rightarrow C_{12}$$

# ADD RI

| CI2 | CI1 | CI0 | C9 | C8 | C7 | C6 | C5 | C4 | … | C0 | BA | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | I | I | 0 | 0 | 0 | 0 | … | 0 | L+I | I |
| I | I | I | 0 | 0 | 0 | 0 | 0 | 0 | … | 0 | 0 0 | I |

# ADD RI

- I need two micro instructions. So it is a program micro program consisting of 2 micro instructions.
- This will be the value of L.
- So L will now be this value which has been generated by the external address generator.
- Next address field will be L + 1 and the M will be equal to 1 and after that next address field will be 0 0, M will also be equal to 1 because the next address field will come from here only.

# Advantage of micro programming

- You find that because this whole thing is loaded in a control memory and the control memory can be programmed as per requirement.

- if I find that today I decide that the control signals are to be generated in some sequence but next day I find that the sequence in which I have generated the control signals they are not proper. I should change control signals the sequence of control signals.

# Control Signal

- if I want to change the sequence of control signals, when I have the hardware control circuit generator then the entire hardware has to be changed.
- speed wise this will be slow because for generation of any control signal, I have to read the memory content. So because it is software in nature, it will be slower than hardware control unit but the advantage is it is more flexible

# Computer Organization

# Computer Organization

# Computer Organization

# Computer Organization