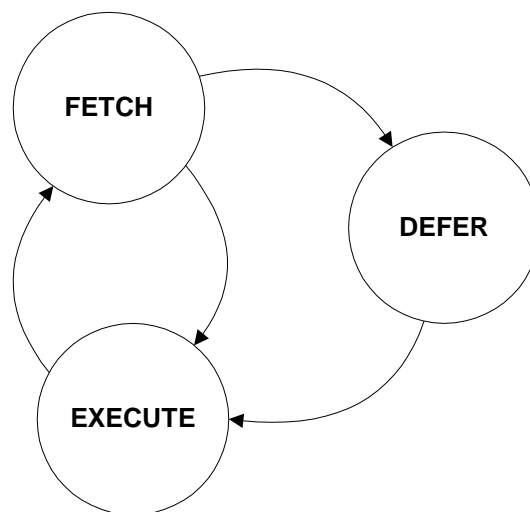


Lecture #16 – Hardwired Control Unit

- Major Instruction Execution Cycles

Each instruction uses up to three major cycles:

1. **FETCH**: Instruction is fetched, PC is incremented
2. **DEFER**: Indirect addressing (if needed)
3. **EXECUTE**: Instruction is executed



- Instruction Fetch

From previous lecture remember that the instruction fetch requires:

- $MAR \leftarrow PC$
- Read Memory
- $IR \leftarrow MBR$
- $PC \leftarrow PC + 1$

Knowing that memory takes some time to respond, we can re-order the transfer as follows to reduce the number of cycles:

T_1	$MAR \leftarrow PC$, Read Memory	
T_2	$PC \leftarrow PC + 1$	(memory is settling here)
T_3	$IR \leftarrow MBR$	

This means we need 3 minor cycles to fetch the instruction into the Instruction Register.

The next two major cycles are either DEFER or EXECUTE. In both of these cycles we need to have the address portion of the instruction loaded into these MAR.

So to save duplication, lets put the loading of the MAR at the end of the FETCH cycle.

The MAR may need to be added to the index register, so the total final minor cycle is:

$$T_4 \quad \text{MAR} \leftarrow \text{IR}_{0-7} + \text{INDEX}$$

- Instruction Fetch Control Signals

Now we need to see how this can be implemented using our bus structure:

$$\begin{array}{ll} T_1 \text{ MAR} \leftarrow \text{PC, Read Memory} & \text{PC to BUS1, TRA1, BUS3 to MAR, READ} \\ T_2 \text{ PC} \leftarrow \text{PC} + 1 & \text{PC to BUS1, 1 to BUS2, ADD, BUS3 to PC} \\ T_3 \text{ IR} \leftarrow \text{MBR} & \text{MBR to BUS2, TRA2, BUS3 to IR} \\ T_4 \text{ MAR} \leftarrow \text{IR}_{0-7} + \text{INDEX} & \text{IR}_{0-7} \text{ to BUS1, INDEX to BUS2, ADD, BUS3 to MAR} \end{array}$$

ALL OF THE CONTROL SIGNALS FOR A MINOR CYCLE ARE ISSUED SIMULTANEOUSLY! THIS MEANS PC to BUS1, ACC to BUS1 IS **NOT** ALLOWED IN THE SAME MINOR CYCLE.

- Defer (Indirect Addressing)

Since the FETCH major cycle has loaded the MAR with the address, if indirect addressing is needed (either regular indirect or indexed indirect), the following is required.

- READ Memory
- $\text{MAR} \leftarrow \text{MBR}$

But remember that memory takes a bit of time to settle, so the minor cycles for DEFER are:

$$\begin{array}{ll} T_1 & \text{READ} \\ T_2 & \text{NOOP} \\ T_3 & \text{MAR} \leftarrow \text{MBR} \end{array}$$

We have already decided that the FETCH major cycle requires 4 minor cycles. To keep things simple we would prefer to have all of the major cycles have the same number of minor cycles.

Since we cannot think of anything better for the DEFER to do, just wait a bit during the last minor cycle.

$$\begin{array}{ll} T_1 & \text{READ} \\ T_2 & \text{NOOP} \\ T_3 & \text{MAR} \leftarrow \text{MBR} \\ T_4 & \text{NOOP} \end{array}$$

- Instruction Fetch Control Signals

Now we need to see how this can be implemented using our bus structure:

T ₁ READ	READ
T ₂ NOOP	
T ₃ MAR ← MBR	MBR to BUS2, TRA2, BUS3 to MAR.
T ₄ NOOP	

Remember this cycle will only get activated if the instruction has indirect addressing.

- Instruction Execute

The EXECUTE major cycle will be different for each instruction.

For the LDA instruction:

T ₁ READ	READ
T ₂ NOOP	
T ₃ ACC ← MBR	MBR to BUS2, TRA2, BUS3 to ACC
T ₄ NOOP	

For the ADD instruction:

T ₁ READ	READ
T ₂ NOOP	
T ₃ ACC ← MBR + ACC	MBR to BUS2, ACC to BUS1, ADD, BUS3 to ACC
T ₄ NOOP	

For the TCA instruction:

T ₁ ACC = <u>ACC</u>	ACC to BUS1, COMP, BUS3 to ACC
T ₂ ACC = ACC + 1	ACC to BUS1, 1 to BUS2, ADD, BUS3 to ACC
T ₃ NOOP	
T ₄ NOOP	

For the TDX instruction:

T ₁ INDEX = INDEX - 1	INDEX to BUS2, -1 to BUS1, ADD, BUS3 to INDEX
T ₂ IF ZERO-INDEX THEN PC = MAR	IF ZERO-INDEX THEN MAR to BUS1, TRA1, BUS3 to PC
T ₃ NOOP	
T ₄ NOOP	

- Zero Address Instructions

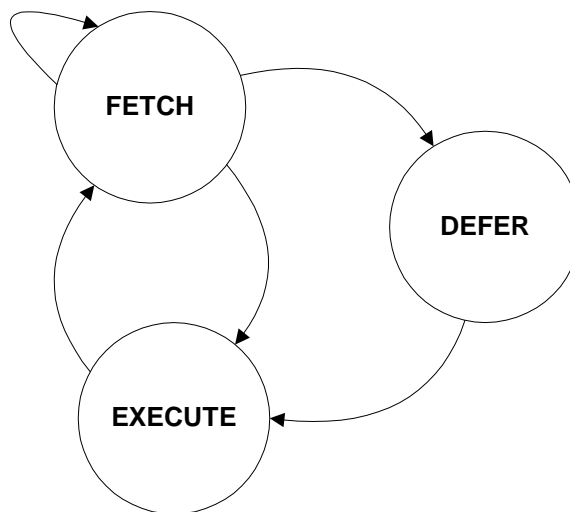
There are three instructions that do not need an address. They are:

SHR Shift Accumulator Right
 SHL Shift Accumulator Left
 HLT Halt

We could simply implement these as instructions with one minor cycle in EXECUTE and a NOOP in the other three minor cycles.

But the last minor cycle of FETCH is not need either. An optimization can be performed with a bit more logic to put the execution of these instructions in the last cycle of FETCH.

Now we get a state diagram as follows:



Using our previous understanding of synchronous sequential circuits we can implement this state diagram using two flip-flops and some combinational logic.

Assign states values:

	D ₁	D ₀
Fetch	0	0
Defer	0	1
Execute	1	0

Therefore:

$D_0 = IR_{10} = 1$ and the current state is 00 (FETCH).

$D_1 = (\text{NOT}(\text{SHL or SHR or HLT}) \text{ and current state} = 00) \text{ OR current state} = 01$ (DEFER).

- Circuits

Generator for Minor Cycles:

