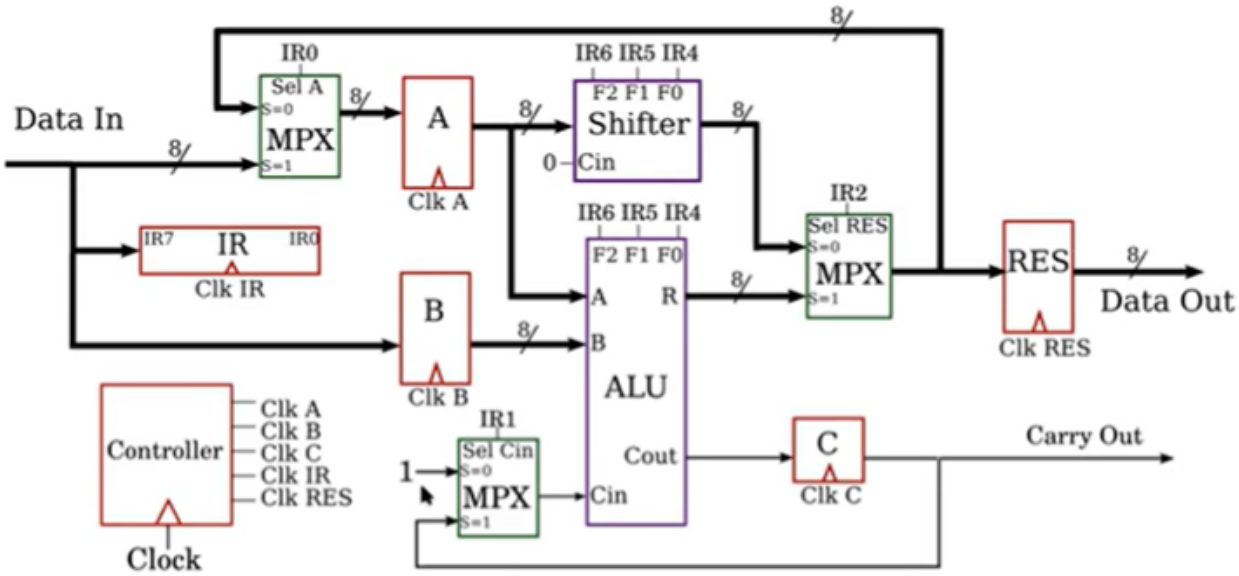
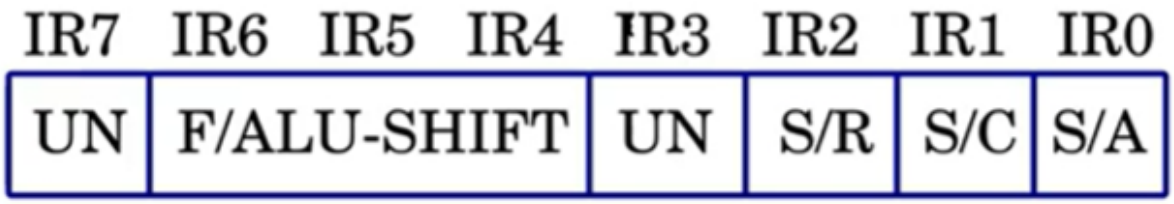
L16 - A Manual Processor (Part 2)





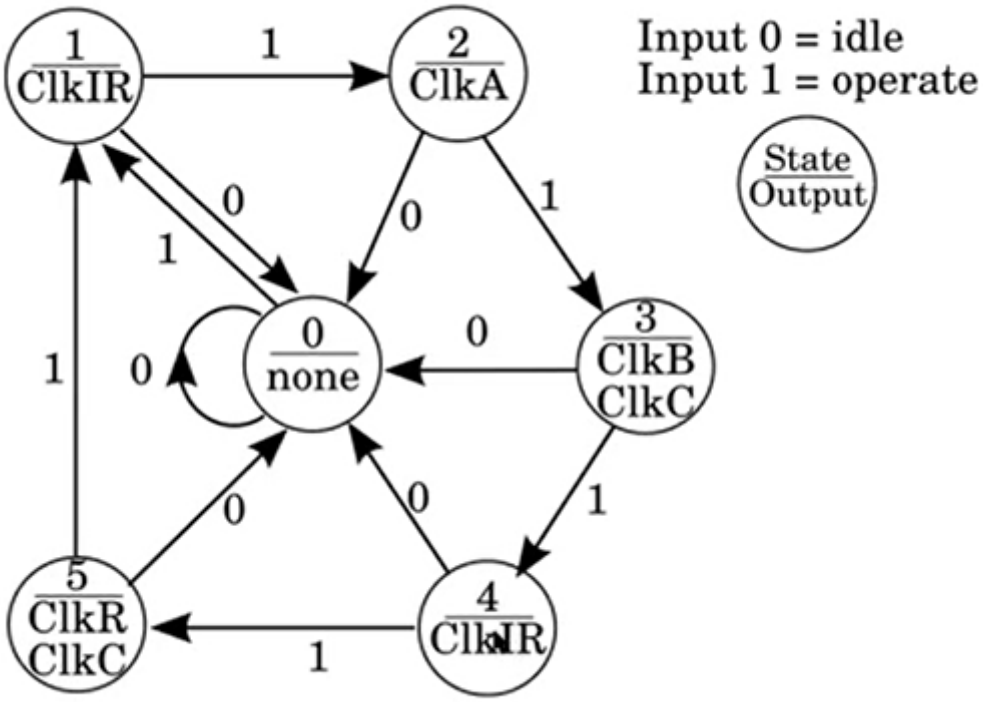
The Instruction Register IR

Now we must program the circuit so that it does the correct function. To do this, we must design the controller. An instruction will take 5 steps to execute:

1. Load the IR Register (to program where to load A from (the Data In path, or from inside the circuit (Accumulation)))
2. Load the A Register
3. Load the B and C registers (A and B must be loaded separately since there is only one Data In line) (Load C to prepare for instruction)
4. Load the IR register (to load the instruction we want)
5. Load the RES and the C registers (the results of the computation)
   Go back to step 1 and repeat

We assume the Data Input lines will hold the correct byte at each step.

To implement this cycle, we need a 5 state synchronous sequential circuit:



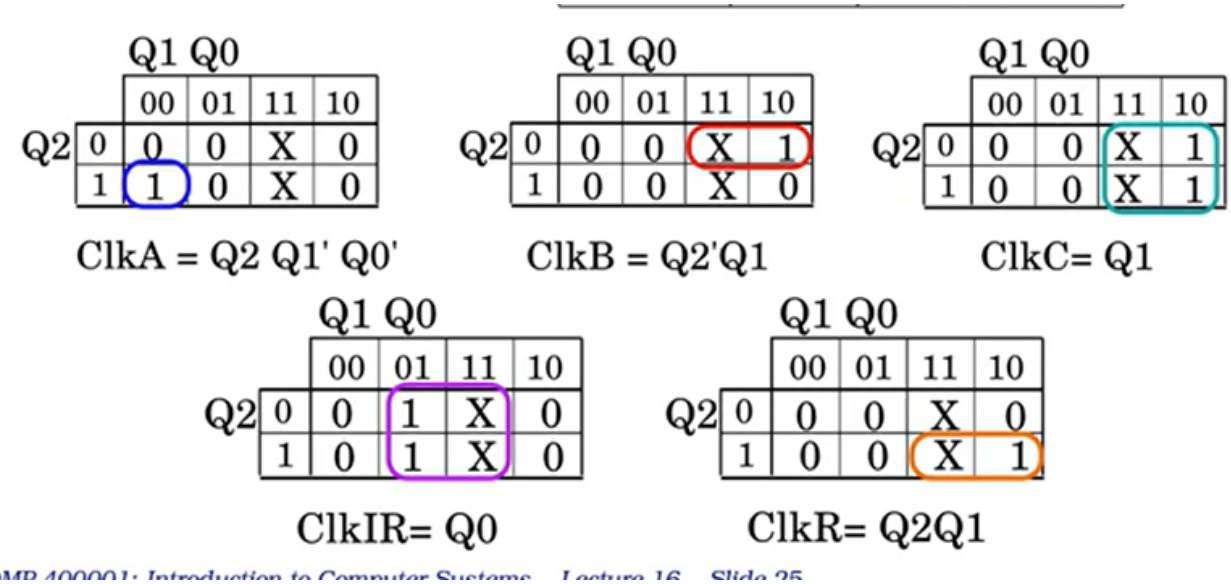
This execution cycle defines the format of the program. At each of the five steps, the correct data will be placed on the Data Input lines.

The Execute Cycle:

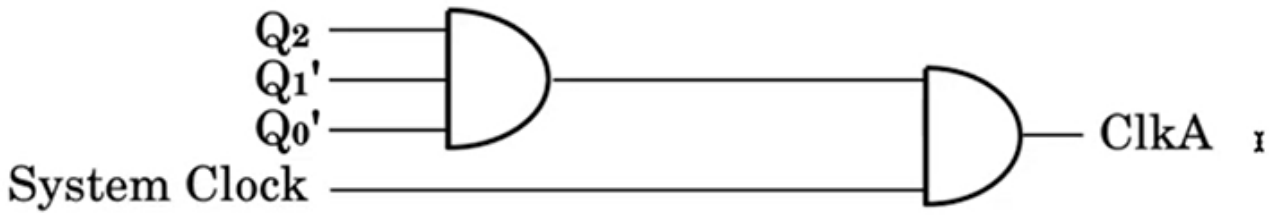
1. Load IR from Data In lines
2. Load Register A (e.g. from Data In)
3. Load Registers B and C
4. Load IR again from Data In
5. Load Registers RES and C
6. Cycle is Complete, restart from (1) in next instruction
7. Load IR from Data In lines
8. Load Register A (e.g. from result of ALU)
   etc

Output Logic
Choose State assignments to minimise the output logic.

$Q_2Q_1Q_0$	State	Output Signal
000	0	none
001	1	ClkIR
100	2	ClkA
010	3	ClkB, ClkC
101	4	ClkIR
110	5	ClkC, ClkRES

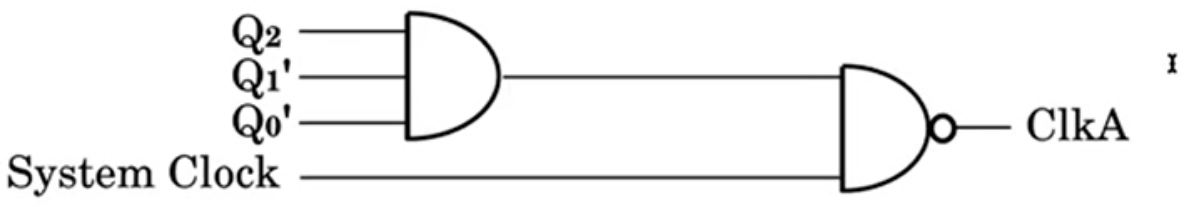


The OL provides a clock edge to the registers that are to be loaded at each step. This can be done by gating the system clock which is timing each step of the execution. E.g.



This arrangement would mean that the controller state register and the processor registers change on the same clock edge - creating a race hazard.

Instead,



To make sure the circuit operates correctly, we make the state register change on the falling edge of the clock and the processor registers change on the next rising edge of the clock. Now, no race hazard.

State Transition Table

Operate	State Now	$Q_2Q_1Q_0$	Next State	D_2	D_1	D_0
0	0	000	0	0	0	0
0	1	001	0	0	0	0
0	2	100	0	0	0	0
0	3	010	0	0	0	0
0	4	101	0	0	0	0
0	5	110	0	0	0	0
0	6	011	×	×	×	×
0	7	111	×	×	×	×
1	0	000	1	0	0	1
1	1	001	2	1	0	0
1	2	100	3	0	1	0
1	3	010	4	1	0	1
1	4	101	5	1	1	0
1	5	110	1	0	0	1
1	6	011	×	×	×	×
1	7	111	×	×	×	×

Checking Don't Cares

Operate	State Now	$Q_2Q_1Q_0$	Next State	D_2	D_1	D_0
0	6	011	0	0	0	0
0	7	111	0	0	0	0
1	6	011	4	1	0	1
1	7	111	4	1	0	1

Thus, if OPERATE input is 0, the system immediately drops into the IDLE state.
The processor is ready to start working properly straight afterwards.
If OPERATE input is 1, the system goes into state 4.
Therefore, we should make sure we start with OPERATE 0 or not in states 6 or 7.

Find mean of two numbers:
Two execute cycles:

0) Set OPERATE input to 0

1) put processor in IDLE state

1. Set Data In lines to x000xxx1 and the OPERATE input to 1
    1. Instruction is loaded into IR
    2. ALU output and Cout are set to 0
    3. Set (Sel A) multiplexer to Data in lines
2. Set Data In lines to Number A
    1. Data In lines are loaded into register A
3. Set Data In lines to Number B
    1. Data In lines are loaded into register B
    2. 0 is loaded into Register C
4. Set the Data In lines to x011x11x
    1. Instruction is loaded into IR
    2. ALU output = A + B
    3. Sel RES multiplexer selects ALU output
    4. Sel C multiplexer selects C register making Cin = 0
5. •
    1. (A + B) loaded into RES register
    2. C bit indicates whether there has been an overflow

2nd Execute Cycle

1. Set Data In lines to x011x110
    1. Instruction is loaded in the IR
    2. ALU output = remains as (A + B)
    3. Sel A and Sel RES multiplexers select the result of the ALU
2. •
    1. ALU output (A + B) loaded into register A
3. •
    1. Register B is loaded with whatever is on the Data In lines (does not matter, we are not using it)
    2. C is loaded with the ALU/Cout again
4. Set the Data In lines to x110x0xx
    1. The instruction is loaded in the IR

2. The shifter function is set to Arithmetic Right Shift and the Sel RES multiplexer selects the shifter output

5.
```
1) The final result (A+B)/2 is loaded into the RES
register. The carry out is not needed for this
instruction.
```

   Set Operate = 0 to put processor back in the idle state.