ICS-LectureReview3

Attention

For the detail of all instructions, please read **Appendix A**

For the complete Finite State Machine of all instructions, please read **Appendix C**.

If still have questions, ask the TAs for further explanation.

Chapter 3 -- Digital Logic Structure

More about Finite State Machine

1. Moore and Mealy Model

Moore Model:
$next\ state = F_1(current\ state, Input), Output = F_2(current\ state)$

Mealy Model:
$next\ state = F_1(current\ state, Input), Output = F_2(current\ state, Input)$

2. Master-Slave Flip-flop

What we want: We hope the states update of the system is **synchronous**, which means that the storage element will be updated only on the clock edge (rising/positive edge or falling/negative edge)

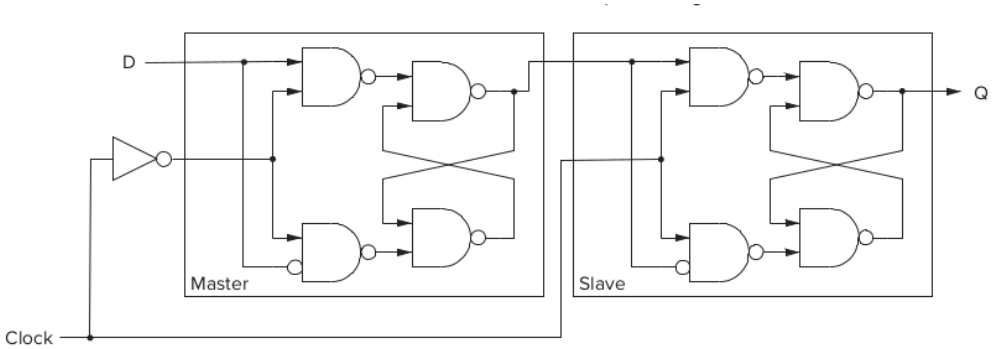
What we have: Gated D-Latch can be written at any time when the WE signal is enabled, which does not meet our needs

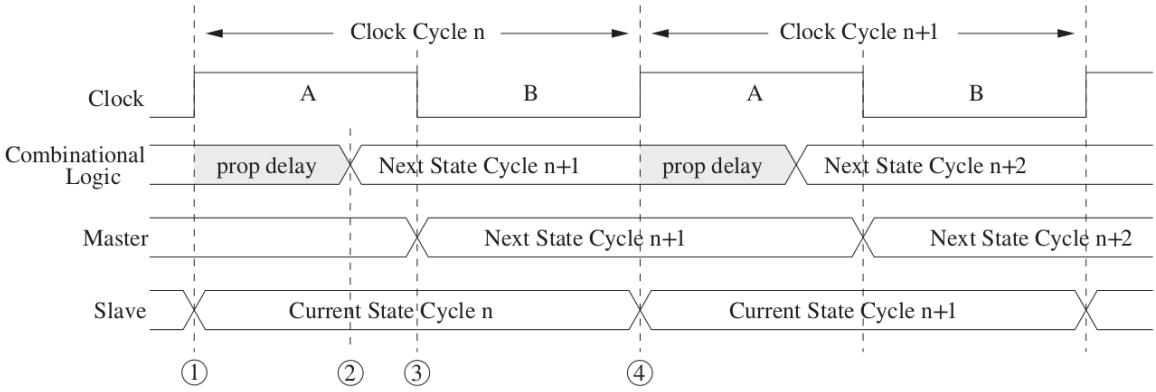
How to solve: Connect the two gated D-latch connected with opposite WE signals ($clock\ and\ \overline{clock}$) together, and the output of the *master* is used as the input of the *slave*, as shown in the figure below.

Why: After the negative edge of the clock, the master can be written, the next state calculated by the combinational logic is written to the master, because the WE (i.e. clock) of the slave is 0, not writable, it still maintains the current state;

At the moment of the positive edge of the clock, the slave is write enabled, the next state stored in the master is written into the slave, which becomes the current state of a new cycle, and the master becomes unwritable afterwards. As a result, we have realized the

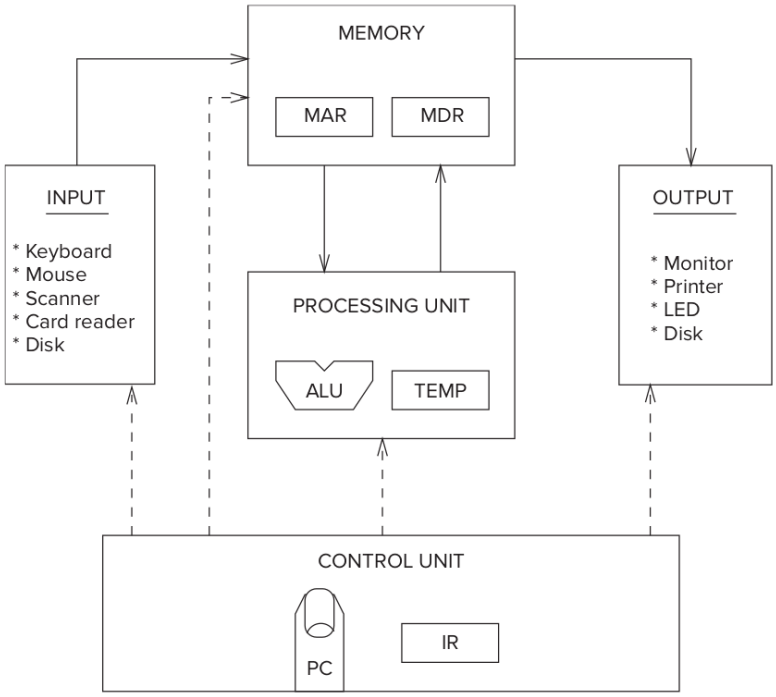
need to update the current state only on the positive edge of the clock.





Chapter 4 -- von Neumann Model

4.1 Basic Components



4.1.1 Memory

4.1.1.1 review

- **address space:** The number of *uniquely identifiable* memory locations
- **addressability:** The number of bits stored in each memory location

4.1.1.2 read from memory

1. load the address you want read to **MAR**, then interrogate the memory
2. the information stored in the location having that address will be placed in the **MDR**

4.1.1.3 write to memory

1. load the address you want write to **MAR**, and the value to be stored in the **MDR**, then interrogate the memory
2. the information contained in the **MDR** will be written into the memory location whose address is in the **MAR**

4.1.2 Processing Unit

Processing unit is the actual unit that carries out the **processing of information** in the computer.

Processing unit can be complex, but now we only focus on the simple case.

4.1.3 Input and Output

Some devices that get input from users, and some display the result to users

4.1.4 Control Unit

It is in charge of making all the other parts of the computer play together. When we describe the step-by-step process of executing a computer program, it is the control unit that **keeps track of both where we are within the process of executing the program and where we are in the process of executing each instruction**.

Control unit in LC-3 consists of **program counter (PC)** and **instruction register (IR)**. PC contains the address of the next instruction, IR contains the current instruction.

4.2 Instruction Processing

4.2.1 The instructions

- **Operation** instructions: operates on data
- **Data movement** instructions: move information from the processing unit to and from memory and to and from input/output devices.
- **Control** instructions: altering the sequential processing of instructions.

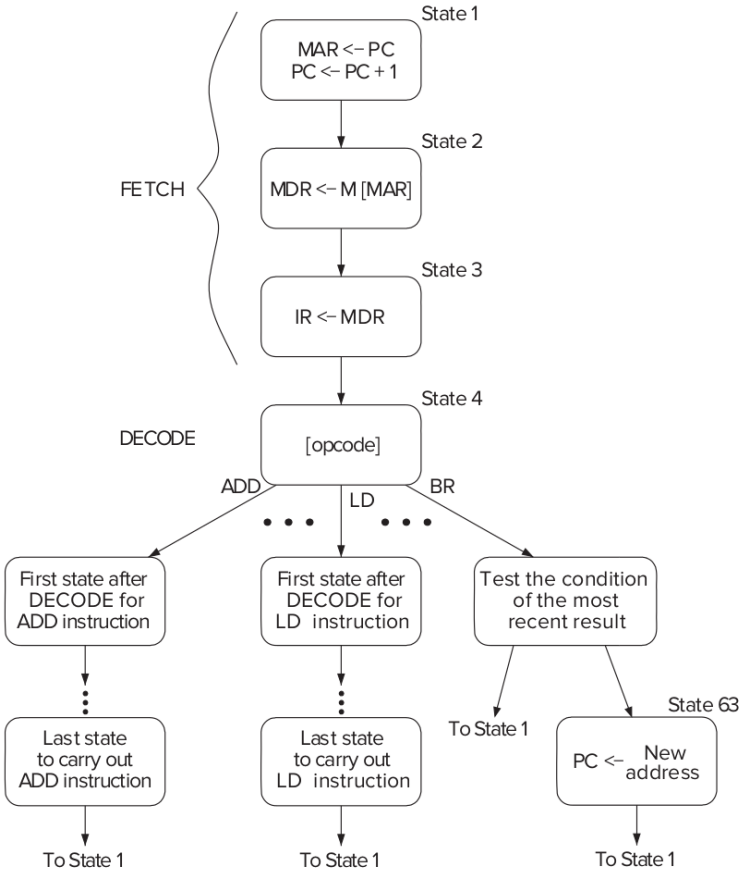
4.2.2 Instruction cycle

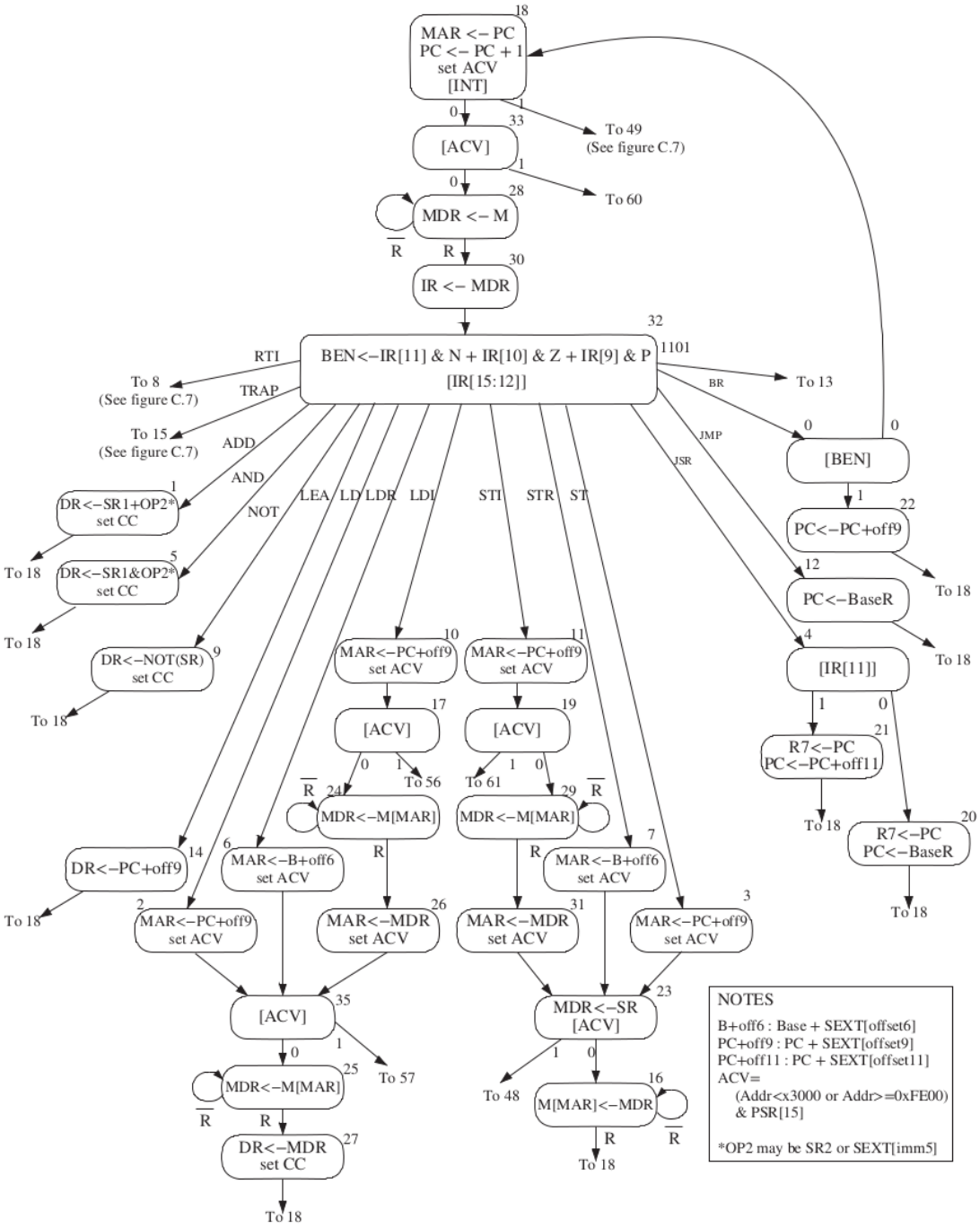
There are **6** stages in LC-3 instruction cycle: **Fetch, Decode, Evaluate address, Fetch operands, Execute, Store result**.

1. **FETCH:** obtains the next instruction from memory and loads it into the instruction register (IR) of the control unit
2. **DECODE:** examines the instruction in order to figure out what the microarchitecture is being asked to do
3. **EVALUATE ADDRESS:** computes the address of the memory
4. **FETCH OPERANDS:** obtains the source operands needed to process the instruction.
5. **EXECUTE:** carries out the execution of the instruction

6. **STORE RESULT:** the result is written to its designated destination (register or memory)

4.2.3 The control of instruction cycle




18
MAR <– PC
PC <– PC + 1
set ACV
[INT]
0
1
To 49
(See figure C.7)
33
[ACV]
0
1
To 60
28
MDR <– M
R
R
30
IR <– MDR
32
BEN<–IR[11] & N + IR[10] & Z + IR[9] & P
[IR[15:12]]
1101
RTI
To 8
(See figure C.7)
TRAP
To 15
(See figure C.7)
ADD
AND
NOT
LEA
LD
LDR
LDI
STI
STR
ST
BR
To 13
JMP
JSR
0
0
[BEN]
1
22
PC<–PC+off9
To 18
12
PC<–BaseR
To 18
4
[IR[11]]
1
0
21
R7<–PC
PC<–PC+off11
To 18
20
R7<–PC
PC<–BaseR
To 18
1
DR<–SR1+OP2*
set CC
To 18
5
DR<–SR1&OP2*
set CC
To 18
9
DR<–NOT(SR)
set CC
To 18
10
MAR<–PC+off9
set ACV
11
MAR<–PC+off9
set ACV
17
[ACV]
0
1
To 56
19
[ACV]
1
0
To 61
R
24
MDR<–M[MAR]
R
29
MDR<–M[MAR]
R
14
DR<–PC+off9
To 18
6
MAR<–B+off6
set ACV
7
MAR<–B+off6
set ACV
2
MAR<–PC+off9
set ACV
26
MAR<–MDR
set ACV
31
MAR<–MDR
set ACV
3
MAR<–PC+off9
set ACV
35
[ACV]
0
1
To 57
23
MDR<–SR
[ACV]
1
0
To 48
25
MDR<–M[MAR]
R
R
16
M[MAR]<–MDR
R
R
27
DR<–MDR
set CC
To 18
To 18
NOTES
B+off6 : Base + SEXT[offset6]
PC+off9 : PC + SEXT[offset9]
PC+off11 : PC + SEXT[offset11]
ACV=
(Addr<x3000 or Addr>=0xFE00)
& PSR[15]
*OP2 may be SR2 or SEXT[imm5]