2013 Fall CSE140L

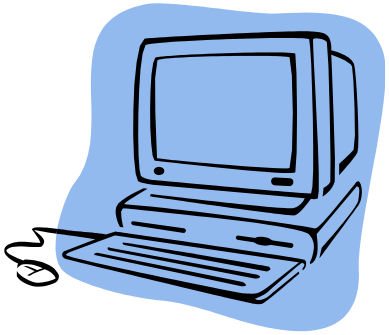
Digital Systems Laboratory

Lecture #8

by

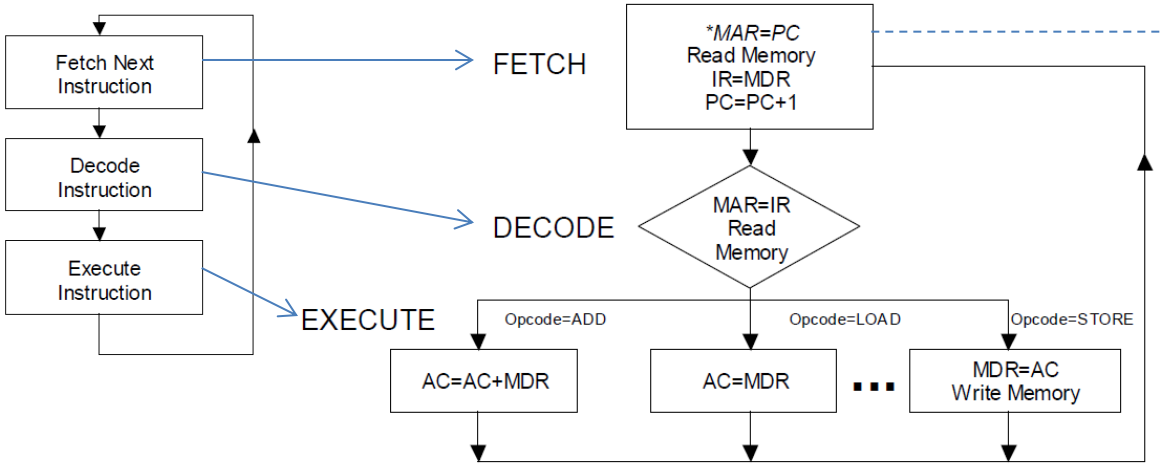
Dr. Choon Kim
CSE Department, UCSD

Practical Sequential Logic Design
(Small computer/CPU example)

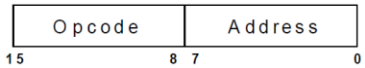


LAB4_tinycpu.pdf

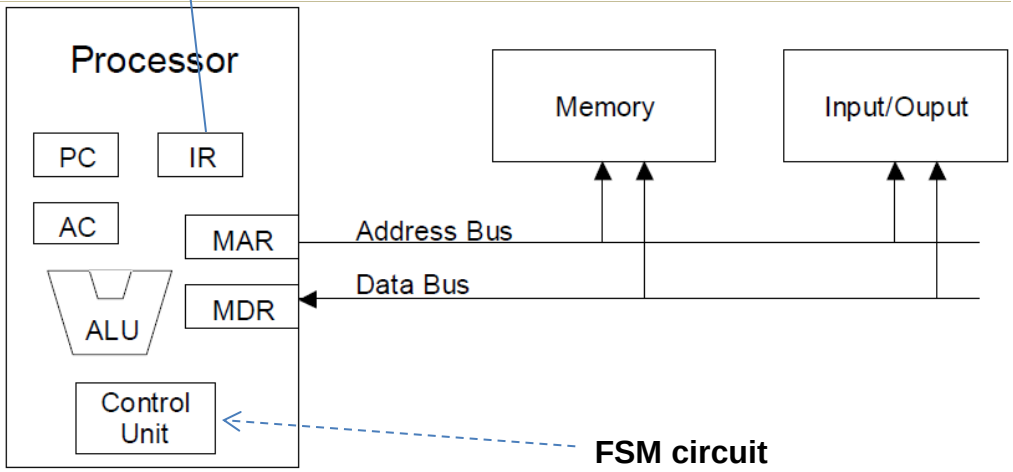
Our LAB4 computer system(16-bit) operation



MAR = PC
Read Memory
MDR = Instruction value from memory
IR = MDR
PC = PC + 1



Instruction	Mnemonic	Operation Preformed	Opcode Value
ADD	*Address*	*AC <= AC + contents of memory Address*	*00*
STORE	*Address*	*contents of memory Address <= AC*	*01*
LOAD	*Address*	*AC <= contents of memory Address*	*02*
JUMP	*Address*	*PC <= Address*	*03*



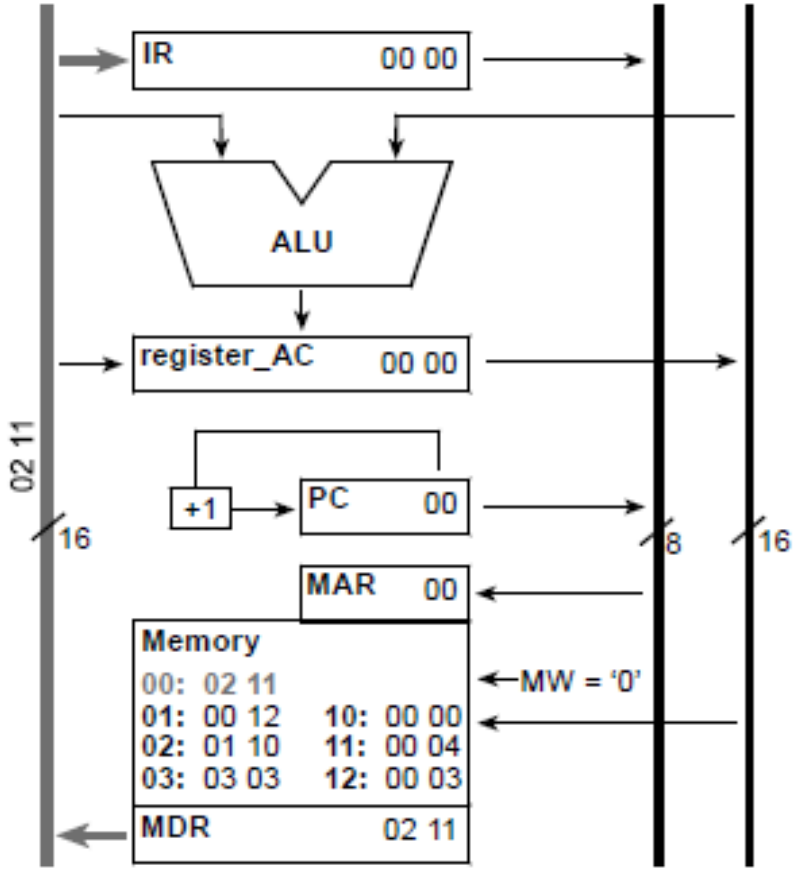
Example Computer Program for A = B + C:

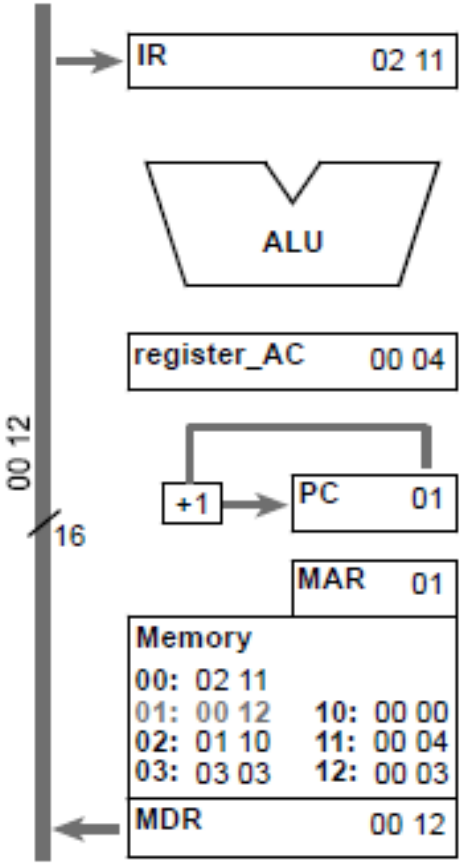
Assembly Language	Machine Language
LOAD B	0211
ADD C	0012
STORE A	0110

Computer Data flow through Datapath

Example Computer Program for A = B + C:

Assembly Language	**Machine Language**
LOAD B	0211
ADD C	0012
STORE A	0110

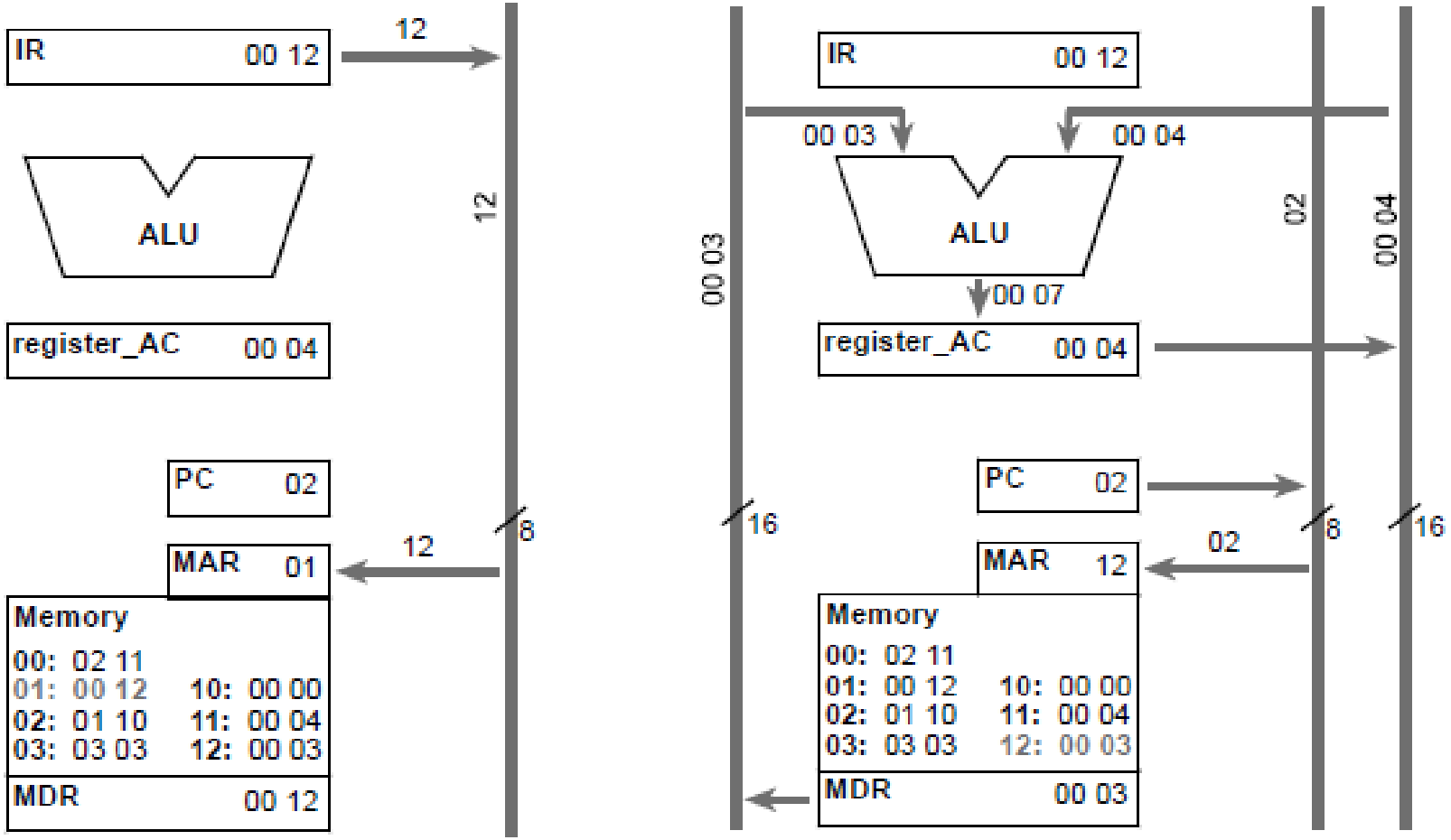




Computer Data flow through Datapath(cont'd)

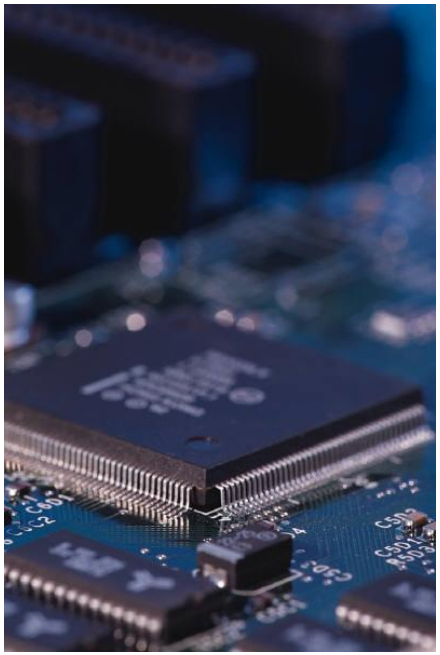
Example Computer Program for A = B + C:

Assembly Language	**Machine Language**
LOAD B	0211
ADD C	0012
STORE A	0110



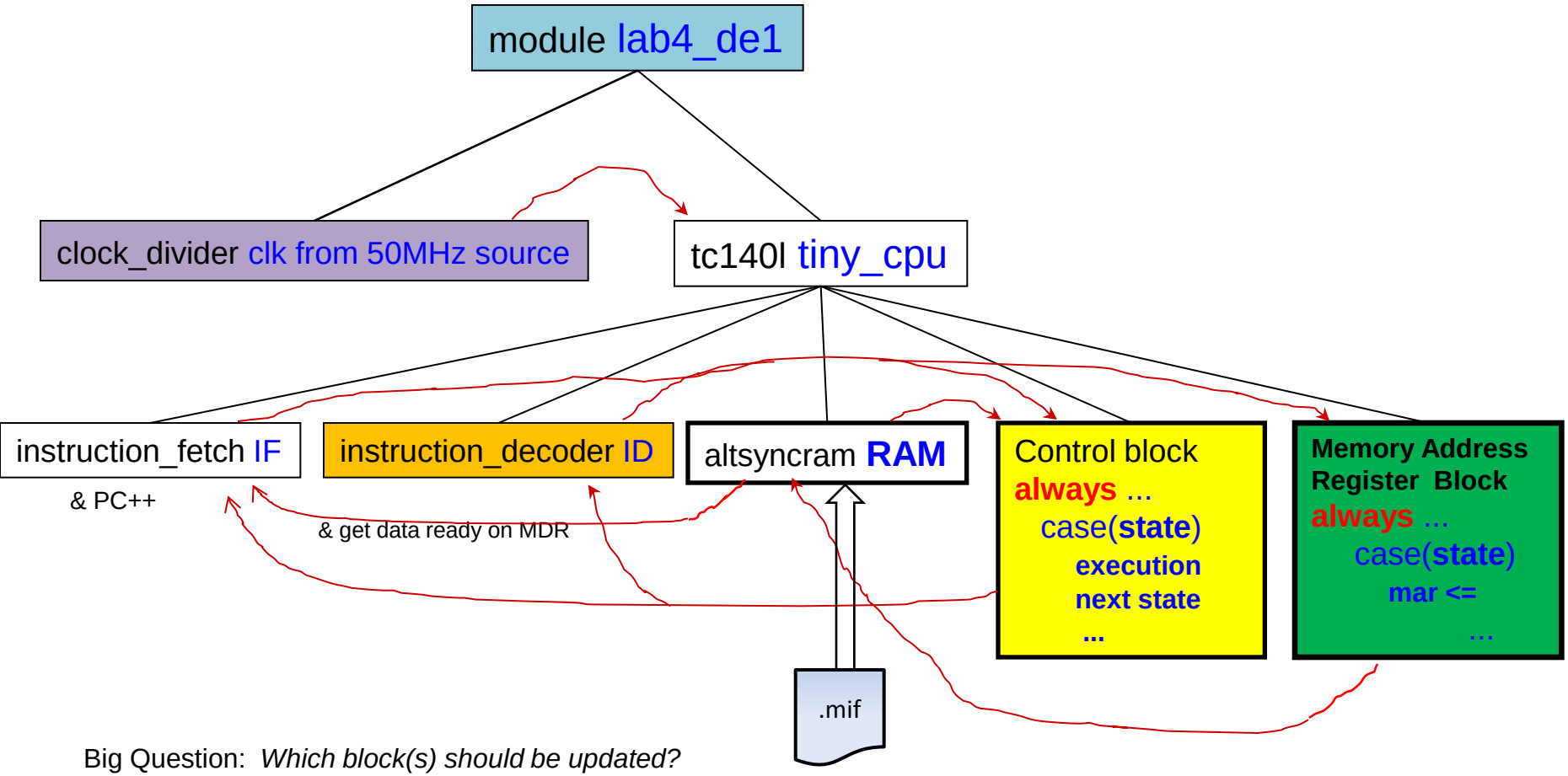
Small Computer Design

Review of LAB#4 Project



LAB4.pdf

LAB4_base project flow review



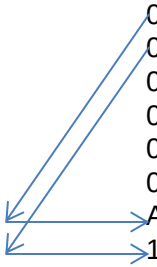
Big Question: *Which block(s) should be updated?*

.mif example

(Example only. May or may not be same as the one in LAB#4)

```
DEPTH = 256;            % Memory depth and width are required      %
WIDTH = 16;             % Enter a decimal number      %

ADDRESS_RADIX = HEX;    % Address and value radixes are optional    %
DATA_RADIX = HEX;       % Enter BIN, DEC, HEX, or OCT; unless       %
                        % otherwise specified, radixes = HEX        %
-- Specify values for addresses, which can be single address or range
-- program   add  A + B
CONTENT
          BEGIN
[00..FF]                : 0000;     % Range--Every address from 00 to FF = 0000 (Default) %
                                    % Warning: Comments may or may not be correct. You must confirm %
                                    % each instruction with definition. %
          00 :          0211;       % LOAD Acc with MEM(11)  %
          01 :          0012;       % Acc = Acc + MEM(12) %
          02 :          0C00;       % OUT Acc %
          03 :          0113;       % STORE Acc to MEM(13) %
          04 :          0300;       % JUMP to 00 (loop forever) %
          10 :          0000;
          11 :          AAAA;
          12 :          1111;
          13 :          0000;
          END ;
```



Question: *What is output pattern of Accumulator when above .mif file is executed?*

Random Number Generation Instruction

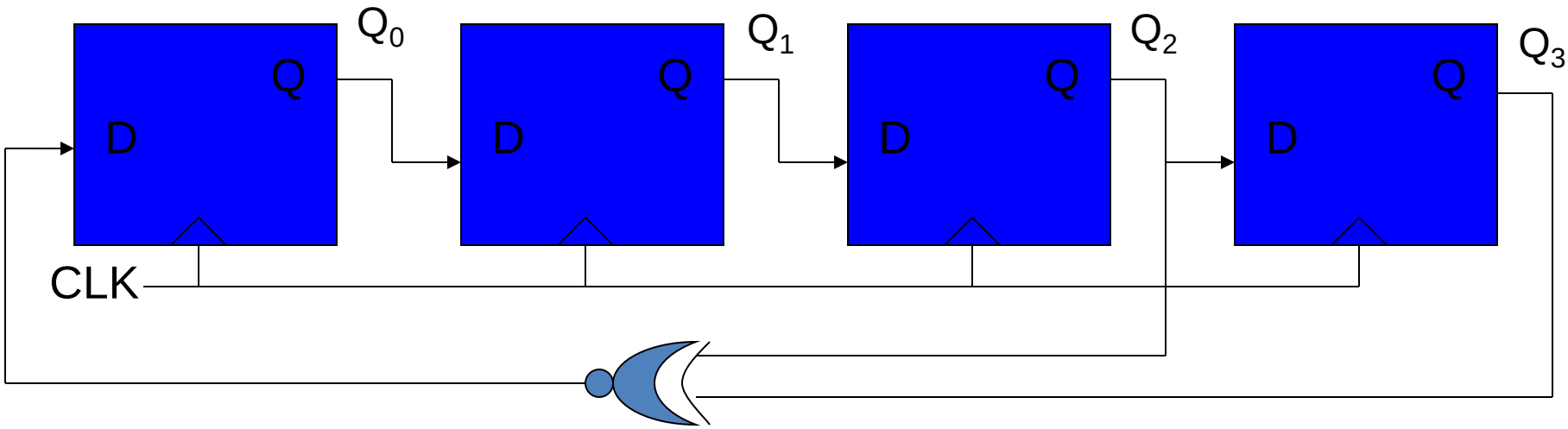
(Example only. May or may not be same as the one in LAB#4)

```
DEPTH = 256;              % Memory depth and width are required      %
WIDTH = 16;               % Enter a decimal number      %

ADDRESS_RADIX = HEX;      % Address and value radixes are optional   %
DATA_RADIX = HEX;         % Enter BIN, DEC, HEX, or OCT; unless      %
                          % otherwise specified, radixes = HEX       %
-- Specify values for addresses, which can be single address or range
-- program pseudo random sequencer
CONTENT
          BEGIN
[00..FF]                  : 0000;     % Range--Every address from 00 to FF = 0000 (Default) %
                                      % Warning: Comments may or may not be correct! You must confirm %
                                      % each instruction with definition in Verilog source code. %
          00 :            3500;       % Random Number Generator instruction  %
          01 :            0300;       % JUMP to 00 (loop forever)  %
          02 :            0000;
          END ;
```

Pseudo Random Sequencer

$D_0 = Q_3$ XNOR Q_2



n = 4, length = 15

4-bit output($Q_3Q_2Q_1Q_0$) is a random number,
where, Q_3 = MSB, Q_0= LSB

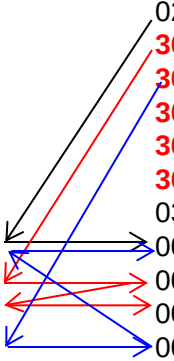
Example of Memory Indirect Addressing Mode

(Example only. May or may not be same as the one in LAB#4)

```
DEPTH = 256;            % Memory depth and width are required        %
WIDTH = 16;             % Enter a decimal number       %

ADDRESS_RADIX = HEX;    % Address and value radixes are optional      %
DATA_RADIX = HEX;       % Enter BIN, DEC, HEX, or OCT; unless         %
                        % otherwise specified, radixes = HEX          %
-- Specify values for addresses, which can be single address or range
-- program addind
CONTENT
          BEGIN
[00..FF]                : 0000;     % Range--Every address from 00 to FF = 0000 (Default) %
                                    % Warning: Comments may or may not be correct! You must confirm %
                                    % each instruction with definition in Verilog source code. %
          00:           0210;       % LOAD AC with MEM(10) -- initialize AC %
          01:           3611;       % ADD IND memory contents to AC %
          02:           3613;       % ADD IND memory contents to AC %
          03:           3611;       % ADD IND memory contents to AC %
          04:           3613;       % ADD IND memory contents to AC %
          05:           3611;       % ADD IND memory contents to AC %
          06:           0300;       % JUMP to 00 (loop forever)     %
          10:           0002;
          11:           0012;
          12:           0009;
          13:           0010;
          14:           0005;

          END ;
```



Example of Memory PC Reference Addressing Mode

(Example only. May or may not be same as the one in LAB#4)

```
DEPTH = 256;            % Memory depth and width are required      %
WIDTH = 16;             % Enter a decimal number      %

ADDRESS_RADIX = HEX;    % Address and value radixes are optional    %
DATA_RADIX = HEX;       % Enter BIN, DEC, HEX, or OCT; unless       %
                        % otherwise specified, radixes = HEX        %
-- Specify values for addresses, which can be single address or range
-- program addpcr
CONTENT
            BEGIN
[00..FF]    :       0000;   % Range--Every address from 00 to FF = 0000 (Default)      %
            00:     0210;   % LOAD AC with Mem(10) -- initialize AC %
            01:     3710;   % ADD PCR memory content to AC %
            02:     3712;   % ADD PCR memory content to AC %
            03:     3714;   % ADD PCR memory content to AC %
            04:     0300;   % JUMP to 00 (loop forever) %
            10:     0009;
            11:     0001;
            12:     0005;
            13:     0003;
            14:     0004;
            15:     0007;
            16:     0006;
            17:     0007;
            18:     000B;
            END ;
```