

2014 Fall CSE140L

Digital Systems Laboratory

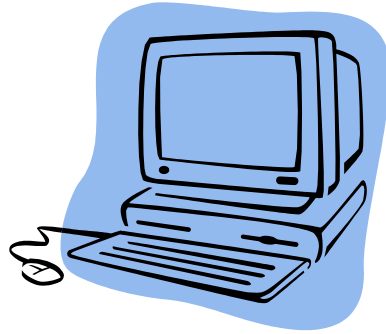
Lecture #8910

by

Dr. Choon Kim
CSE Department, UCSD

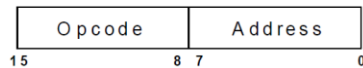
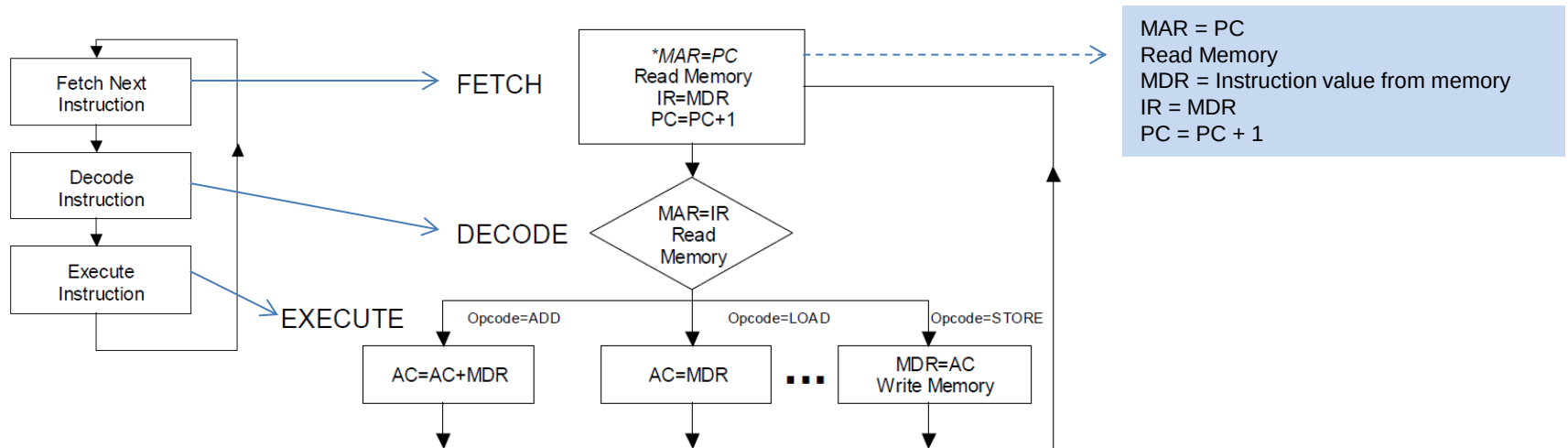
Practical Sequential Logic Design

(Small computer/CPU example)

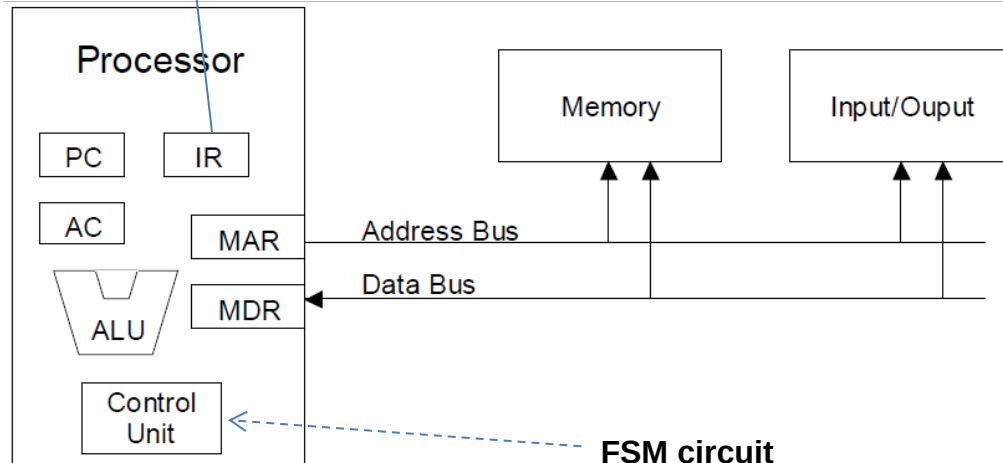


[LAB4_tinycpu.pdf](#)

Our LAB4 computer system(16-bit) operation



Instruction	Mnemonic	Operation Performed	Opcode Value
ADD	Address	$AC \leq AC + \text{contents of memory Address}$	00
STORE	Address	$\text{contents of memory Address} \leq AC$	01
LOAD	Address	$AC \leq \text{contents of memory Address}$	02
JUMP	Address	$PC \leq \text{Address}$	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

LOAD B

ADD C

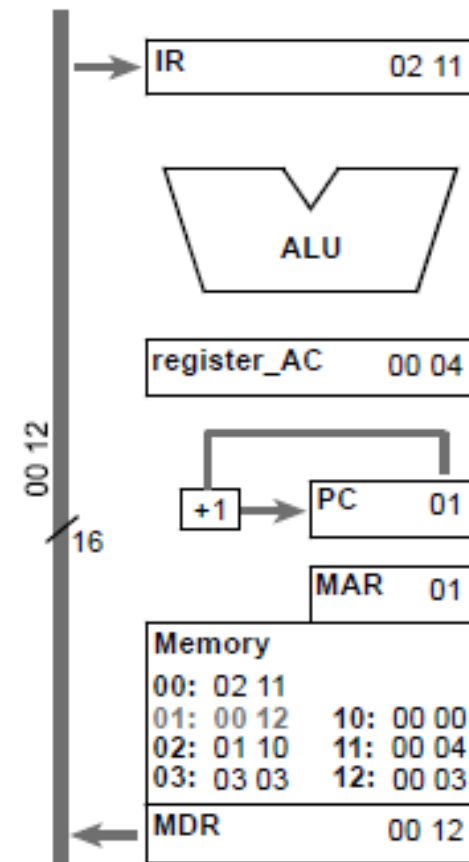
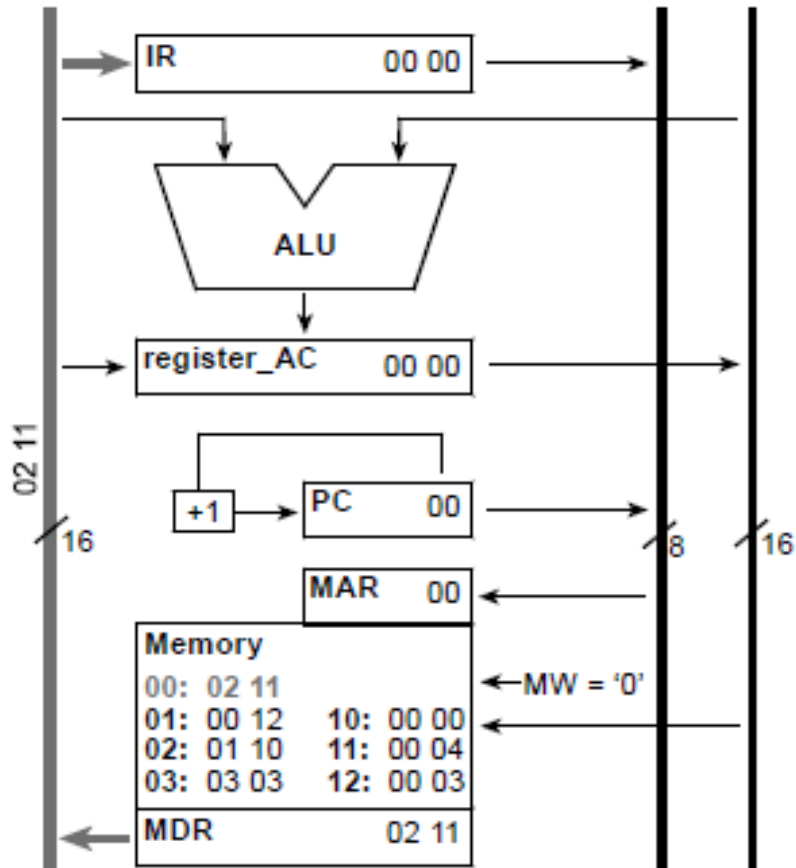
STORE A

Machine Language

0211

0012

0110



Computer Data flow through Datapath(cont'd)

Example Computer Program for $A = B + C$:

Assembly Language

LOAD B

ADD C

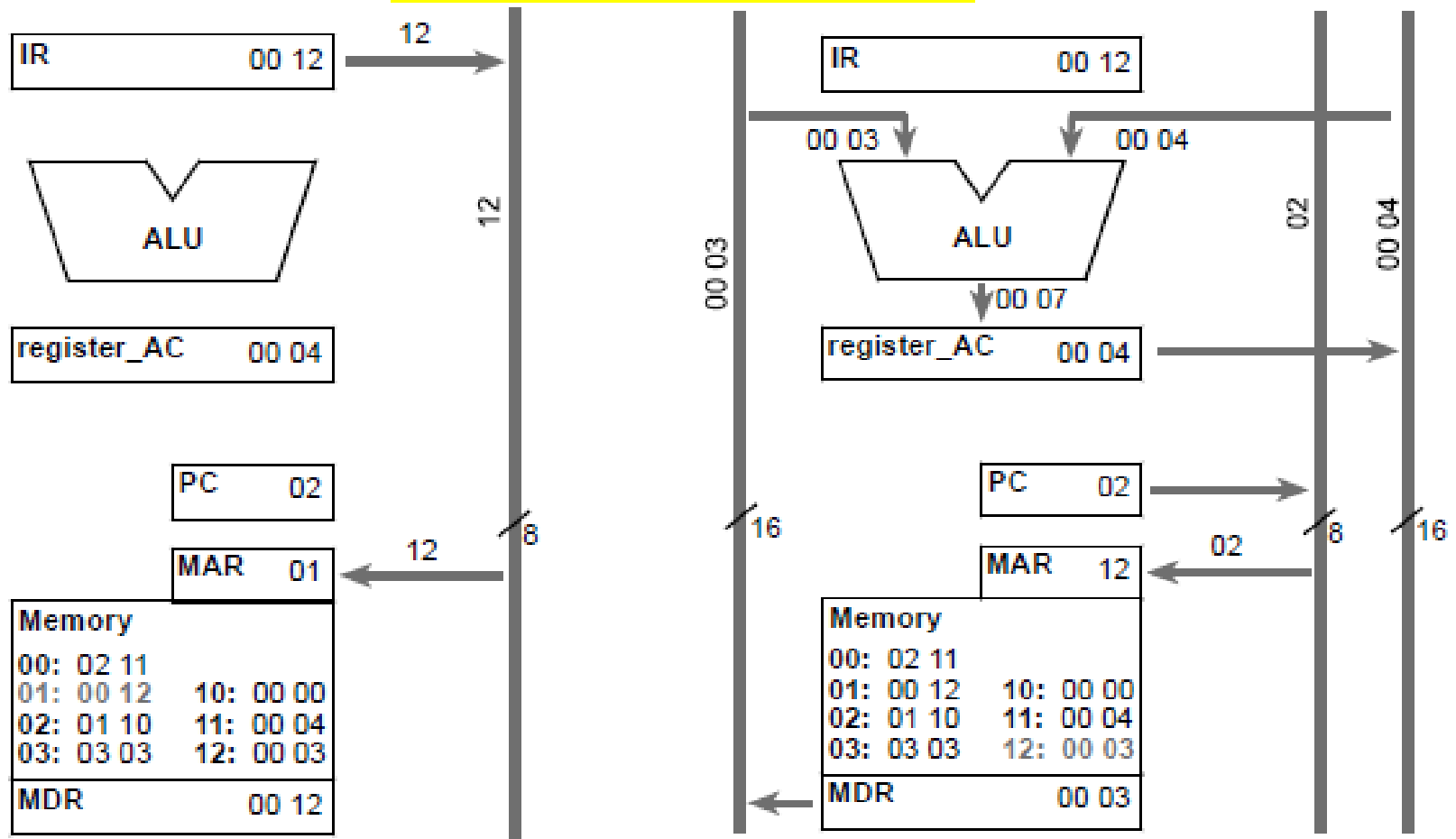
STORE A

Machine Language

0211

0012

0110



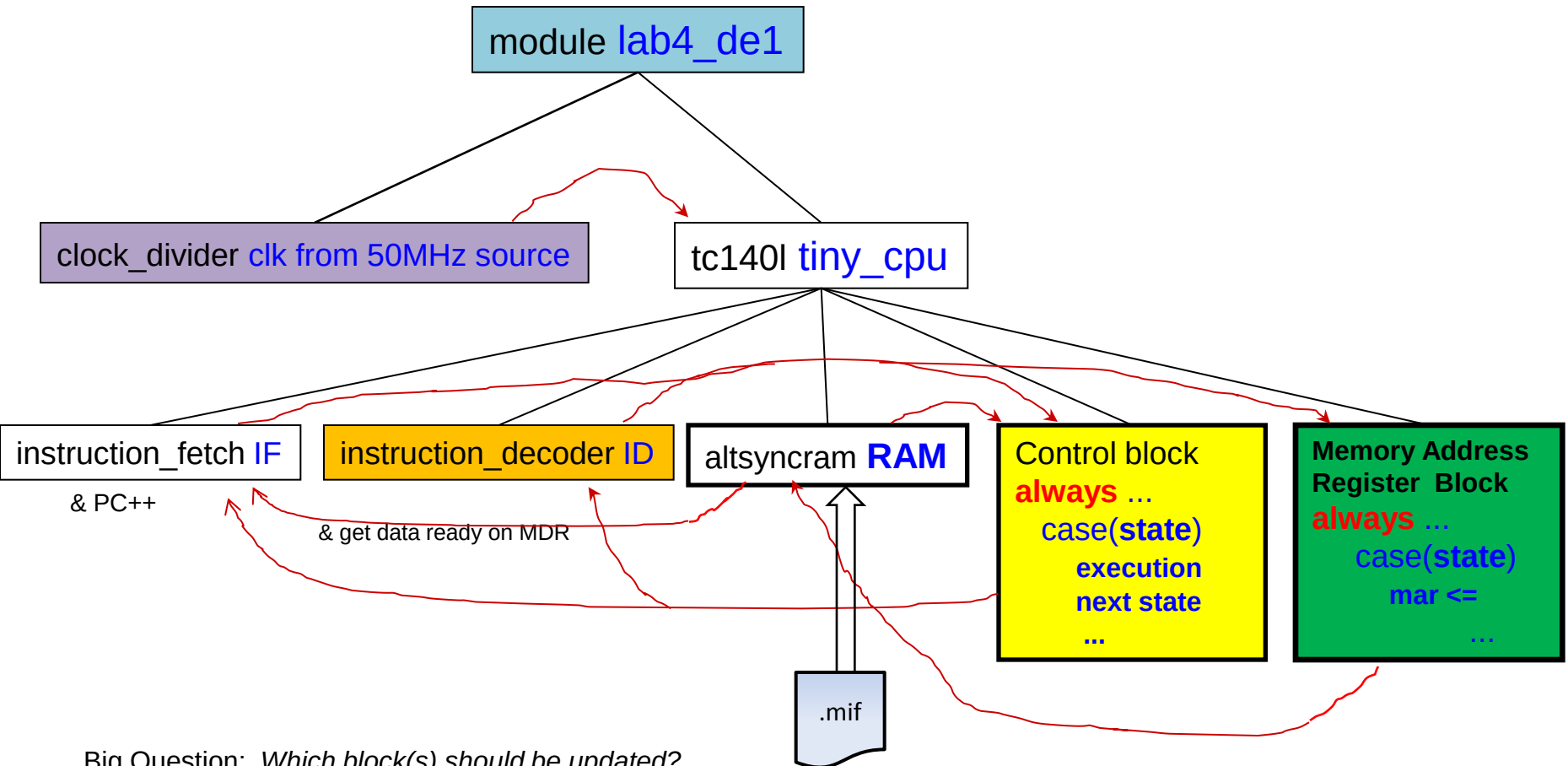
Small Computer Design

Review of LAB#4 Project



[LAB4.pdf](#)

LAB4_base project flow review



.mif example

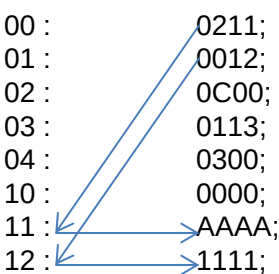
(Example only. May or may not be same as the one in LAB#4)

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

ADDRESS_RADIX = HEX;   % Address and value radices are optional   %
DATA_RADIX = HEX;      % Enter BIN, DEC, HEX, or OCT; unless      %
                      % otherwise specified, radices = 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. %

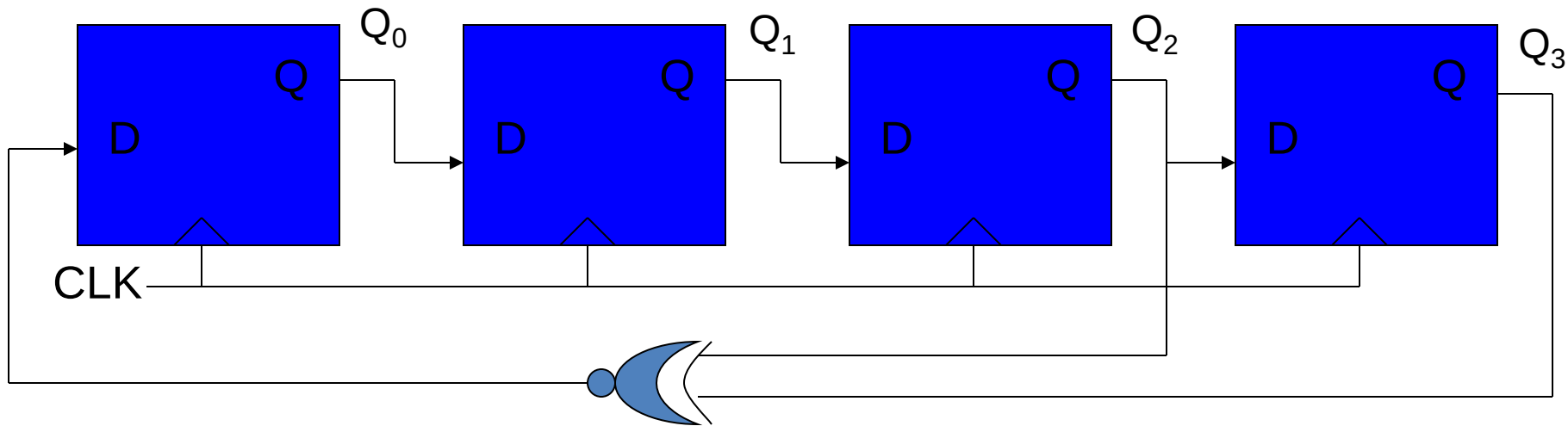
    00 :                   0211;    % LOAD Acc with MEM(11) %
    01 :                   0012;    % Acc = Acc + MEM(12) %
    02 :                   0C00;    % OUT Acc %
    03 :                   0113;    % STORE Acc to MEM(13) %
    04 :                   0300;    % JUMP to 00 (loop forever) %
    10 :                   0000;
    11 :                   AAAA;
    12 :                   1111;
    13 :                   0000;
    END ;
```



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

Pseudo Random Sequencer

$$D_0 = Q_3 \text{ XNOR } Q_2$$



$n = 4$, length = 15

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

Random Number Generation Instruction("random.mif")

(Example only! 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 radices are optional  %
DATA_RADIX = HEX;        % Enter BIN, DEC, HEX, or OCT; unless    %
                        % otherwise specified, radices = 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 : ← 5500;                % LOAD AC with Random number %
    01 : 0300;                  % JUMP to 00 (loop forever) %
    10 : 0000;
    11 : 5555;
    12 : 1111;
    13 : 0000;
    END ;
```

Example of Memory Indirect Addressing Mode (Example only! 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 radices are optional  %
DATA_RADIX = HEX;      % Enter BIN, DEC, HEX, or OCT; unless     %
                        % otherwise specified, radices = 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:                   5611;    % ADD IND memory contents to AC %
    02:                   5613;    % ADD IND memory contents to AC %
    03:                   5611;    % ADD IND memory contents to AC %
    04:                   5613;    % ADD IND memory contents to AC %
    05:                   5611;    % 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 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 radices are optional   %
DATA_RADIX = HEX;      % Enter BIN, DEC, HEX, or OCT; unless      %
                        % otherwise specified, radices = HEX       %

-- Specify values for addresses, which can be single address or range
-- program addpcr
CONTENT

[00..FF] BEGIN
:          0000;          % Range--Every address from 00 to FF = 0000 (Default)      %
00:        0210;          % LOAD AC with Mem(10) -- initialize AC %
01:        5710;          % ADD PCR memory content to AC %
02:        5712;          % ADD PCR memory content to AC %
03:        5714;          % 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 ;

```

The diagram illustrates the execution flow of the program. It shows a list of memory addresses from 00 to 18. Arrows indicate jumps between addresses: a black arrow from 04 to 10, a red arrow from 10 to 12, a blue arrow from 12 to 11, a green arrow from 11 to 10, a blue arrow from 10 to 15, a green arrow from 15 to 18, a red arrow from 18 to 17, a blue arrow from 17 to 16, a green arrow from 16 to 15, a red arrow from 15 to 14, a blue arrow from 14 to 13, a green arrow from 13 to 12, a red arrow from 12 to 11, a blue arrow from 11 to 10, and a green arrow from 10 to 04. The values at these addresses are: 00: 0000, 01: 0210, 02: 5710, 03: 5712, 04: 0300, 10: 0009, 11: 0001, 12: 0005, 13: 0003, 14: 0004, 15: 0007, 16: 0006, 17: 0007, 18: 000B.