

# CENG3420 Computer Organization & Design

## Lab 3-1: LC-3b Datapath

**Bei Yu**

Spring 2016

[byu@cse.cuhk.edu.hk](mailto:byu@cse.cuhk.edu.hk)



香港中文大學  
The Chinese University of Hong Kong

# Overview

Introduction

Lab3-1 Assignment

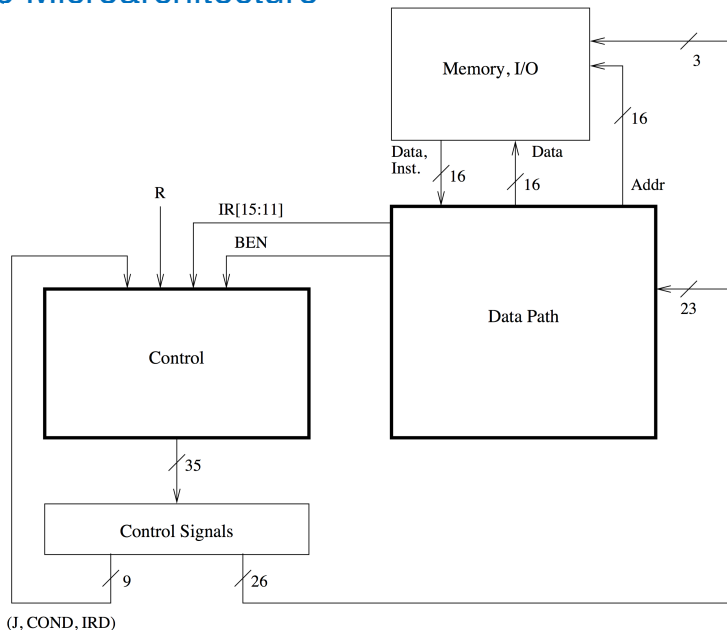
Golden Results

# The Slides are self-contained? NO!

Do please refer to following document:

- ▶ [LC-3b\\_datapath.pdf](#)
- ▶ [LC-3b\\_ISA.pdf](#)

# LC-3b Microarchitecture



# Input of Control Structure (7 bits)

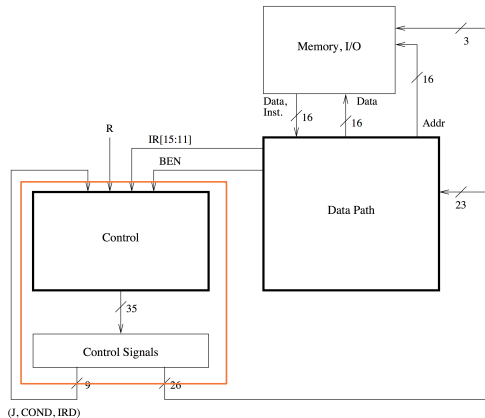
- ▶ **R**: indicate whether memory data is ready (`System_Latches::READY`)
- ▶ **BEN**: indicate whether BR been taken (`System_Latches::BEN`)
- ▶ **IR[15:11]**: current instruction (`System_Latches::IR`)

# Output of Control Structure: (35 bits)

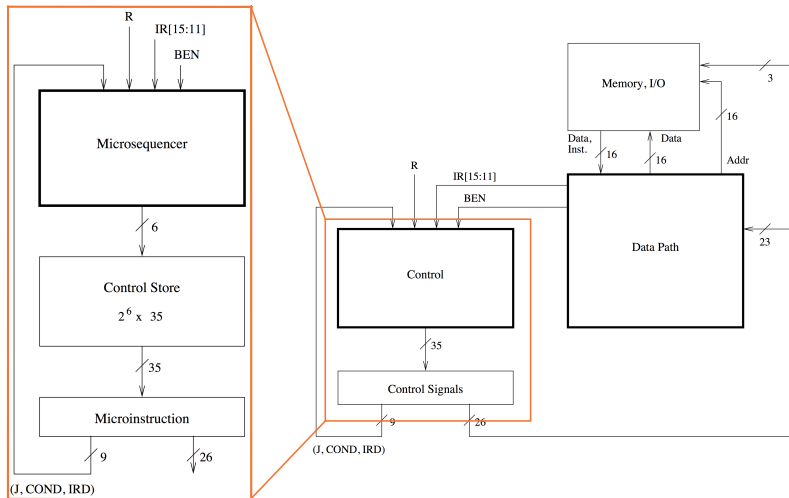
```
45 /******  
46 /* Definition of bit order in control store word.          */  
47 /******  
48 enum CS_BITS {  
49     IRD,  
50     COND1, COND0,  
51     J5, J4, J3, J2, J1, J0,  
52     LD_MAR,  
53     LD_MDR,  
54     LD_IR,  
55     LD_BEN,  
56     LD_REG,  
57     LD_CC,  
58     LD_PC,  
59     GATE_PC,  
60     GATE_MDR,  
61     GATE_ALU,  
62     GATE_MARMUX,  
63     GATE_SHF,  
64     PCMUX1, PCMUX0,  
65     DRMUX,  
66     SR1MUX,  
67     ADDR1MUX,  
68     ADDR2MUX1, ADDR2MUX0,  
69     MARMUX,  
70     ALUK1, ALUK0,  
71     MIO_EN,  
72     R_W,  
73     DATA_SIZE,  
74     LSHF1,  
75     CONTROL_STORE_BITS  
76 } CS_BITS;
```

- ▶ 26 bits to control data path
- ▶ J[5:0], COND[1:0],IRD: generate address of control structure for next clock cycle

# LC-3b Control Structure

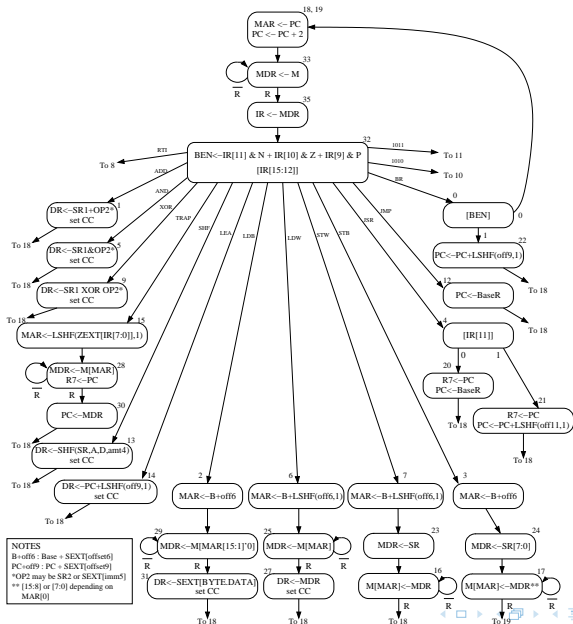


# LC-3b Control Structure





# How's Microsequencer Actually Working?



# How's Control Store Actually Implemented?

000	001	010	011	100	101	110	111	00000 (State 0)
Clear	1	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	00001 (State 1)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	00010 (State 2)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	00011 (State 3)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	00010 (State 4)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	00011 (State 5)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	00010 (State 6)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	00011 (State 7)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	00010 (State 8)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	00011 (State 9)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	00010 (State 10)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	00011 (State 11)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	00010 (State 12)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	00011 (State 13)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	00010 (State 14)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	00011 (State 15)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	01000 (State 16)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	01001 (State 17)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	01010 (State 18)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	01011 (State 19)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	01100 (State 20)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	01101 (State 21)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	01110 (State 22)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	01111 (State 23)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	01000 (State 24)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	01001 (State 25)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	01010 (State 26)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	01011 (State 27)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	01100 (State 28)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	01101 (State 29)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	01110 (State 30)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	01111 (State 31)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	10000 (State 32)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	10001 (State 33)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	10010 (State 34)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	10011 (State 35)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	10100 (State 36)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	10101 (State 37)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	10110 (State 38)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	10111 (State 39)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	10000 (State 40)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	10001 (State 41)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	10010 (State 42)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	10011 (State 43)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	10100 (State 44)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	10101 (State 45)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	10110 (State 46)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	10111 (State 47)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	11000 (State 48)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	11001 (State 49)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	11010 (State 50)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	11011 (State 51)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	11100 (State 52)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	11101 (State 53)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	11110 (State 54)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	11111 (State 55)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	11000 (State 56)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	11001 (State 57)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	11010 (State 58)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	11011 (State 59)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	11100 (State 60)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	11101 (State 61)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	11110 (State 62)
		LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	LD/MAK	11111 (State 63)

## Finite-State-Machine (FSM)

- ▶ States 10, 11 are empty
- ▶ 6 bits input enough
- ▶ Per state, output 35 bits



# Good News!

- ▶ FSM has been provided
- ▶ See file “*ucode3*”

```
107
108 /*****
109 /* The control store rom.          */
110 /*****
111 int CONTROL_STORE[CONTROL_STORE_ROWS][CONTROL_STORE_BITS];
112
```

# Overview

Introduction

Lab3-1 Assignment

Golden Results

# Operations in One Clock Cycle

In "lc3bsim3-1.c":

```
void cycle()
{
    eval_micro_sequencer();
    cycle_memory();
    eval_bus_drivers();
    drive_bus();
    latch_datapath_values();

    CURRENT_LATCHES = NEXT_LATCHES;

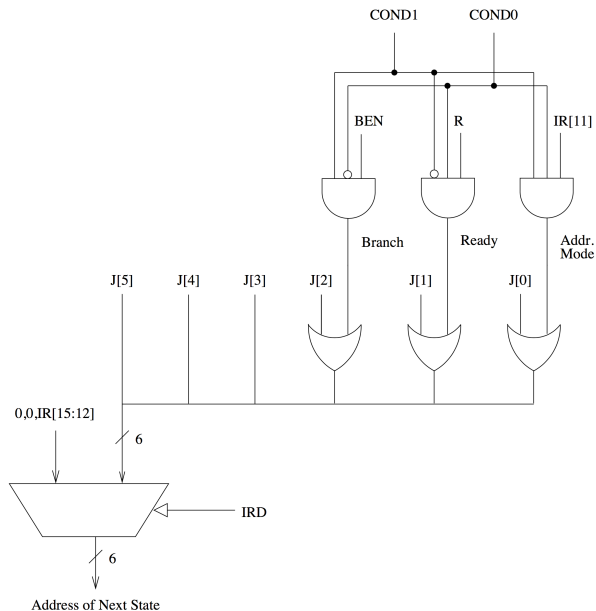
    CYCLE_COUNT++;
}
```

# Lab3-1 Assignment

```
442
443 /*
444  * Evaluate the address of the next state according to the
445  * micro sequencer logic. Latch the next microinstruction.
446  */
447 void eval_micro_sequencer()
448 {
449     /*
450     * Lab3-1 assignment
451     * (the following lines are functionally wrong)
452     */
453
454     NEXT_LATCHES.STATE_NUMBER = 0;
455     memcpy(NEXT_LATCHES.MICROINSTRUCTION, CONTROL_STORE[NEXT_LATCHES.STATE_NUMBER], sizeof(int)*CONTROL_STORE_BITS);
456 }
457
458
```

- ▶ **Input:** CURRENT\_LATCHES
- ▶ **Output:** NEXT\_LATCHES.MICROINSTRUCTION

# Lab3-1 Assignment Tips





## Lab3-1 Assignment Tips (cont.)

Some functions may help:

- ▶ `GetCOND()`
- ▶ `GetIRD()`
- ▶ `GetJ()`
- ▶ `partVal()`

# Overview

Introduction

Lab3-1 Assignment

Golden Results

# Assignment Package

- ▶ `lc3bsim3-1.c`, `lc3bsim3-1.h`: codes to work on
- ▶ `libems3-1.a`: library
- ▶ `ucode3`: FSM
- ▶ `Makefile`
- ▶ `bench`: folder with benchmarks

## Run the simulator:

1. `make`, then binary “`lc3bsim3-1`” is generated
2. `./lc3bsim3-1 ucode3 bench/toupper.cod`

# Golden Results – case `toupper.cod`

## 1. run 6

```
Simulating for 6 cycles...
```

```
MemCycleCnt = 0  
MEM_EN = 0, R_W = 0, WE0 = 0, WE1 = 0  
MemCycleCnt = 0  
MEM_EN = 1, R_W = 0, WE0 = 0, WE1 = 0  
MemCycleCnt = 1  
MEM_EN = 1, R_W = 0, WE0 = 0, WE1 = 0  
MemCycleCnt = 2  
MEM_EN = 1, R_W = 0, WE0 = 0, WE1 = 0  
MemCycleCnt = 3  
MEM_EN = 1, R_W = 0, WE0 = 0, WE1 = 0  
MemCycleCnt = 4  
MEM_EN = 1, R_W = 0, WE0 = 0, WE1 = 0
```

# Golden Results – case toupper.cod (cont.)

## 2. rdump

```
Current register/bus values :
```

```
-----  
Cycle Count   : 6  
PC            : 0x3002  
IR           : 0x0000  
STATE_NUMBER  : 0x0023
```

```
BUS           : 0x0000  
MDR           : 0xe00f  
MAR           : 0x3000  
CCs: N = 0   Z = 1   P = 0
```

```
Registers:
```

```
0: 0x0000  
1: 0x0000  
2: 0x0000  
3: 0x0000  
4: 0x0000  
5: 0x0000  
6: 0x0000  
7: 0x0000
```

# Golden Results – case `toupper.cod` (cont.)

## 3. Go on **run 1**

```
Simulating for 1 cycles...
```

```
MemCycleCnt = 1
```

```
MEM_EN = 0, R_W = 0, WE0 = 0, WE1 = 0
```

# Golden Results – case toupper.cod (cont.)

## 4. rdump

```
Current register/bus values :
```

```
-----  
Cycle Count   : 7  
PC            : 0x3002  
IR            : 0xe00f  
STATE_NUMBER  : 0x0020  
  
BUS           : 0xe00f  
MDR           : 0xe00f  
MAR           : 0x3000  
CCs: N = 0   Z = 1   P = 0  
Registers:  
0: 0x0000  
1: 0x0000  
2: 0x0000  
3: 0x0000  
4: 0x0000  
5: 0x0000  
6: 0x0000  
7: 0x0000
```

# Golden Results – case `toupper.cod` (cont.)

## 5. Go on run 5

Simulating for 5 cycles...

```
MemCycleCnt = 0
MEM_EN = 0, R_W = 0, WE0 = 0, WE1 = 0
MemCycleCnt = 0
MEM_EN = 0, R_W = 0, WE0 = 0, WE1 = 0
MemCycleCnt = 0
MEM_EN = 0, R_W = 0, WE0 = 0, WE1 = 0
MemCycleCnt = 0
MEM_EN = 1, R_W = 0, WE0 = 0, WE1 = 0
MemCycleCnt = 1
MEM_EN = 1, R_W = 0, WE0 = 0, WE1 = 0
```



# Golden Results – case `toupper.cod` (cont.)

## 6. `rdump`

```
Current register/bus values :
```

```
-----  
Cycle Count   : 12  
PC            : 0x3004  
IR           : 0xe00f  
STATE_NUMBER  : 0x0021  
  
BUS           : 0x0000  
MDR           : 0x0000  
MAR           : 0x3002  
CCs: N = 0   Z = 1   P = 0  
Registers:  
0: 0x3020  
1: 0x0000  
2: 0x0000  
3: 0x0000  
4: 0x0000  
5: 0x0000  
6: 0x0000  
7: 0x0000
```

Thanks. For any question:  
[byu@cse.cuhk.edu.hk](mailto:byu@cse.cuhk.edu.hk)