CENG3420
Lab 3-1: LC-3b Datapath

Wei Li

Department of Computer Science and Engineering
The Chinese University of Hong Kong

wli@cse.cuhk.edu.hk

Spring 2020

香港中文大學
The Chinese University of Hong Kong

Overview

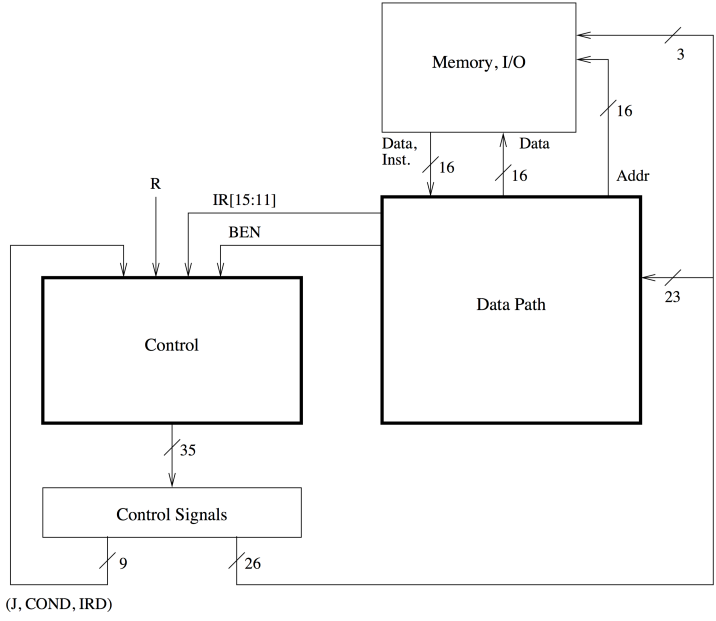
Overview

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)

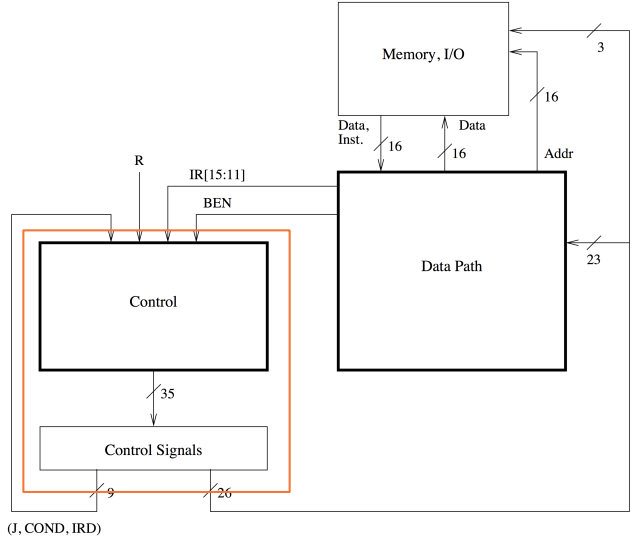
- R: indicate whether memory data is ready (`System_Latches::READY`)
- BEN: indicate whether `BR` been taken (`System_Latches::BEN`)
- IR[15:11]: current instruction (`System_Latches::IR`)

Output of Control Structure: (35 bits)

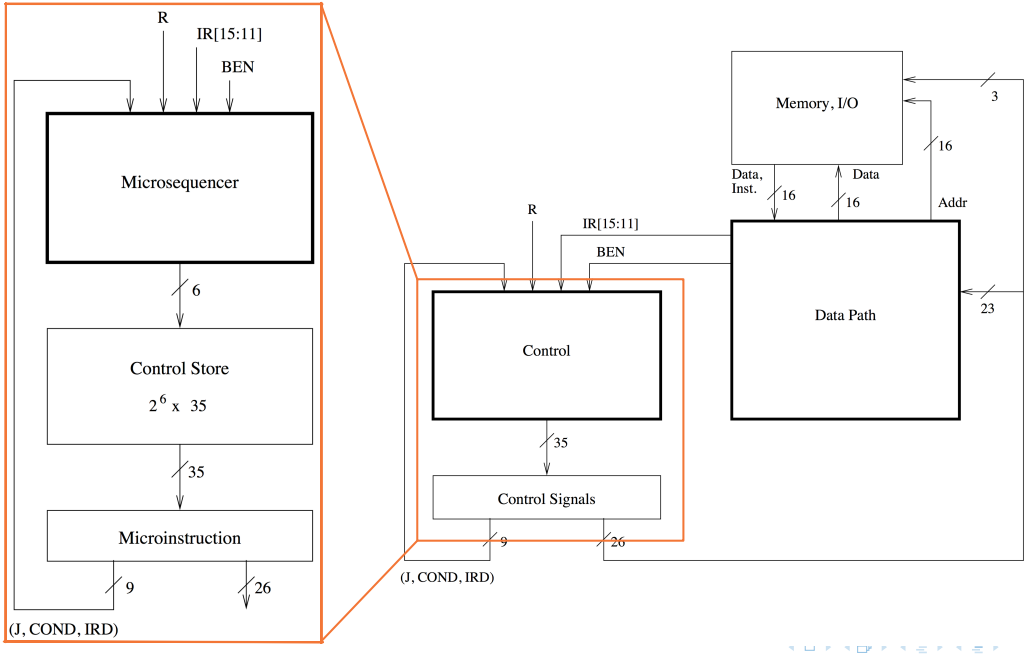
```
44
45 /*****************************************************************/
46 /* Definition of bit order in control store word.                */
47 /*****************************************************************/
48 enum CS_BITS {
49     IRD,
50     COND1, COND0,
51     J5, J4, J3, J2, J1, J0,
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

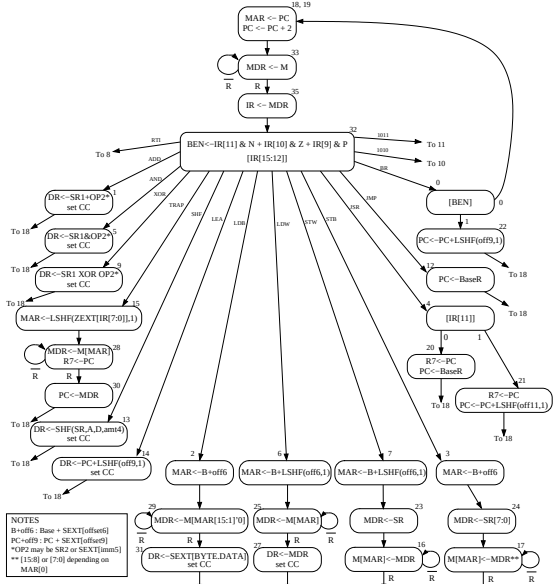
LC-3b Control Structure



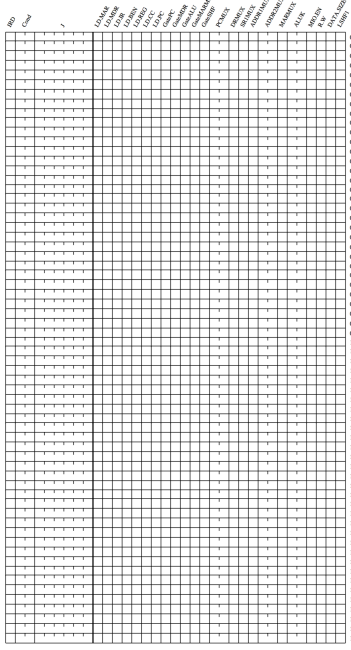
LC-3b Control Structure



How's Microsequencer Actually Working?



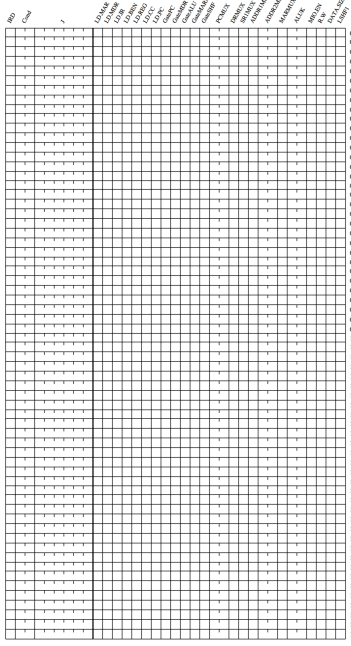
How's Control Store Actually Implemented?



Finite-State-Machine (FSM)

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

How's Control Store Actually Implemented?



Finite-State-Machine (FSM)

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

Hard to implement

Good News!

- Part of 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

Operations in One Clock Cycle

In “lc3bsim3-1.c”:

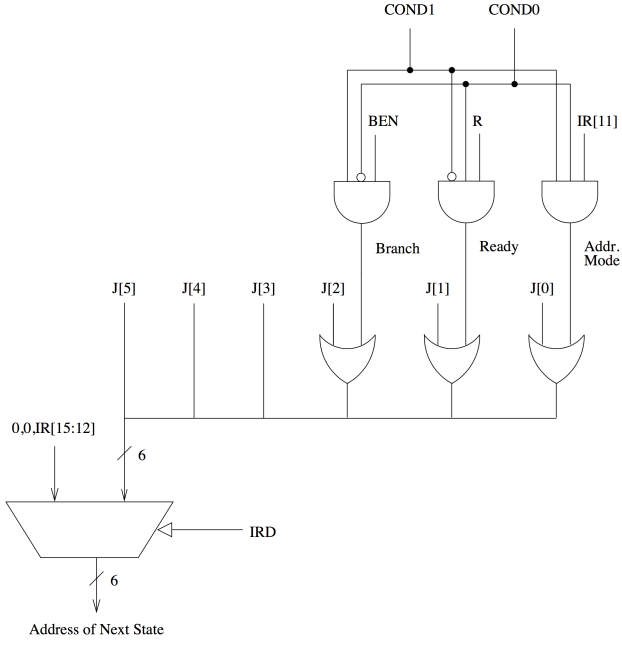
```
void cycle()
{
    eval_micro_sequencer();
    cycle_memory();
    eval_bus_drivers();
    drive_bus();
    latch_datapath_values();

    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      *
452      *  (the following lines are functionally wrong)
453      */
454
455     NEXT_LATCHES.STATE_NUMBER = 0;
456     memcpy(NEXT_LATCHES.MICROINSTRUCTION, CONTROL_STORE[NEXT_LATCHES.STATE_NUMBER], sizeof(int)*CONTROL_STORE_BITS);
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

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 = 0  P = 1
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
wli@cse.cuhk.edu.hk