CENG3420
Lab 3-2: 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

Introduction

Lab3-2 Assignment

Golden Results

Overview

Introduction

Lab3-2 Assignment

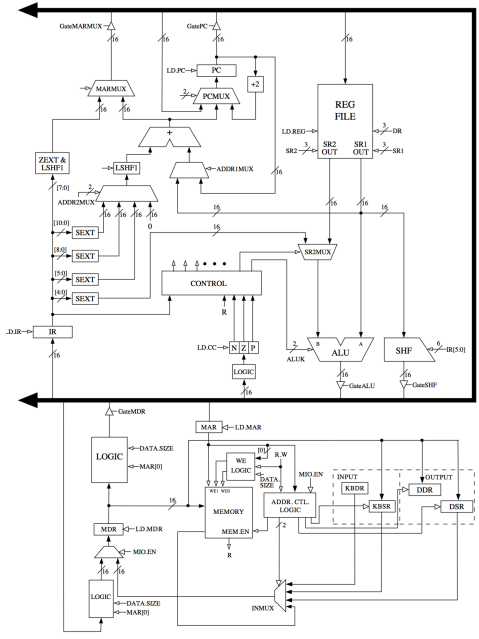
Golden Results

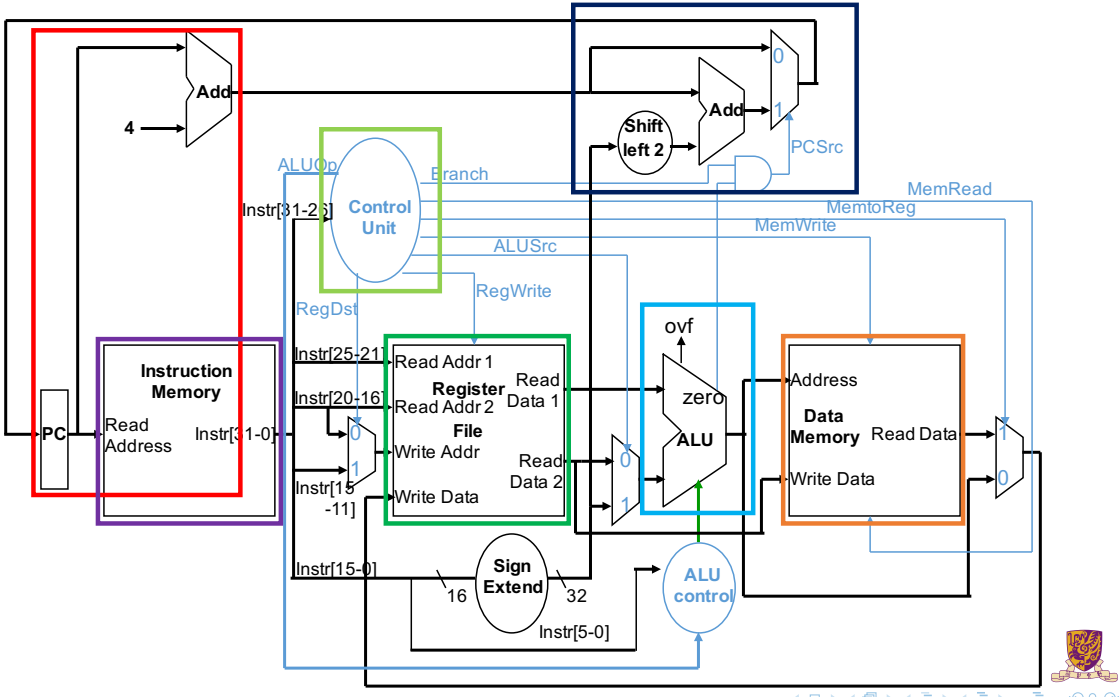
The Slides are self-contained? NO!

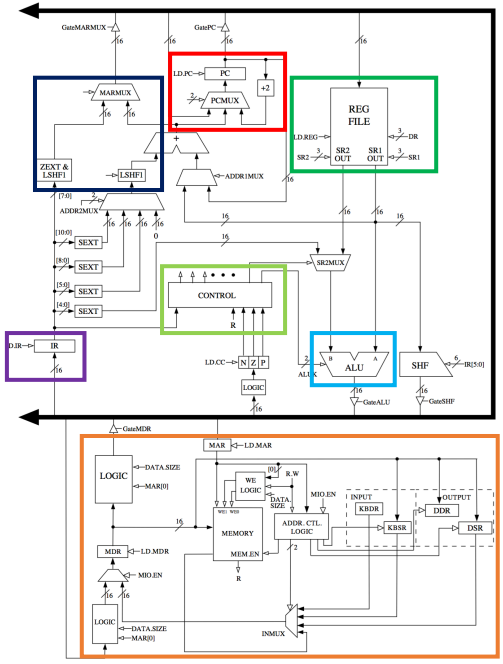
Do please refer to following document:

- LC-3b-datapath.pdf
- LC-3b-ISA.pdf

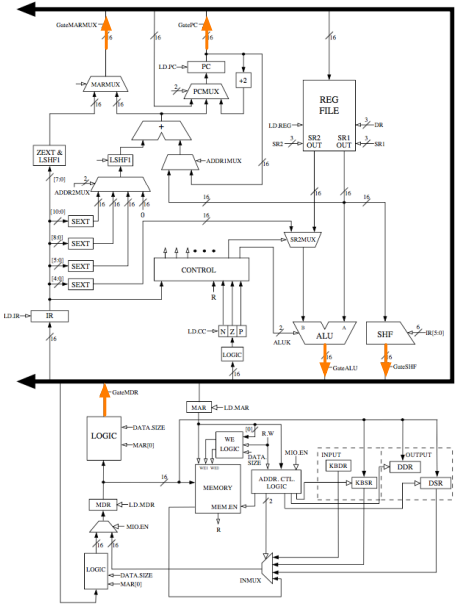
LC-3b Datapath




Add
4
Add
Shift
left 2
0
1
PCSrc
ALUOp
Branch
Control
Unit
Instr[31-26]
MemRead
MemtoReg
MemWrite
ALUSrc
RegWrite
RegDst
ovf
Instruction
Memory
PC
Read
Address
Instr[31-0]
Instr[25-21]
Instr[20-16]
Instr[15
-11]
Instr[15-0]
Read Addr 1
Read Addr 2
Register
File
Write Addr
Write Data
Read
Data 1
Read
Data 2
zero
ALU
Address
Data
Memory
Read Data
Write Data
Sign
Extend
16
32
ALU
control
Instr[5-0]

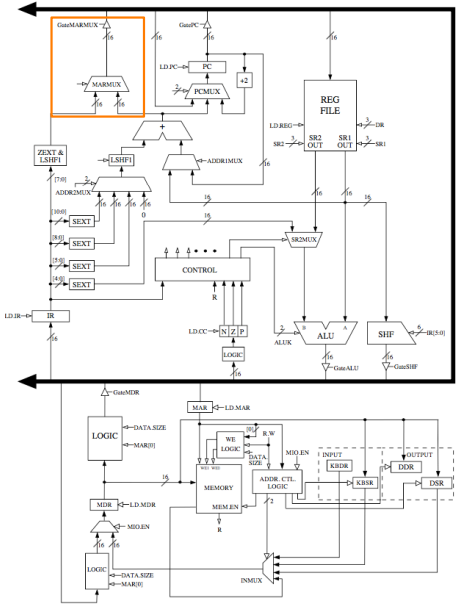

GateMARMUX
GatePC
LD.PC
PC
+2
PCMUX
MARMUX
ZEXT & LSHF1
LSHF1
ADDR1MUX
ADDR2MUX
REG FILE
LD.REG
DR
SR1
SR2
SR2MUX
SEXT
CONTROL
LD.IR
IR
LD.CC
N
Z
P
ALUK
ALU
SHF
IR[5:0]
LOGIC
GateALU
GateSHF
GateMDR
MAR
LD.MAR
DATA.SIZE
MAR[0]
WE
R.W
MIO.EN
INPUT
OUTPUT
KBDR
DDR
KBSR
DSR
MEMORY
ADDR. CTL. LOGIC
MDR
LD.MDR
MEM.EN
INMUX

LC-3b Datapath



- Five major drivers: `MARMUX`, `PC`, `ALU`, `SHF`, `MDR`
- It is the means of data transfer between units

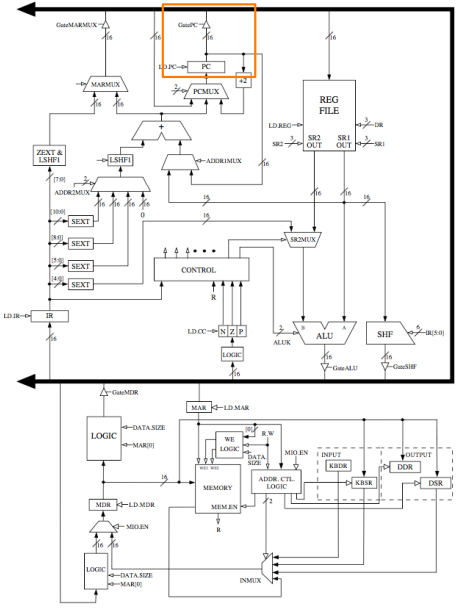
LC-3b Datapath



Driver `MARMUX`

- `int inputOfGateMARMUX;`
- Controlled by `MARMUX`
- from `IR[7:0]` or adder (defined in `inputOfMARMUXFromAdder`)
- `IR[7:0]` is through `LSHF1` (left shift 1 bit) & `ZEXT` (zero extended)
- Implementation of `inputOfMARMUXFromAdder` has been provided

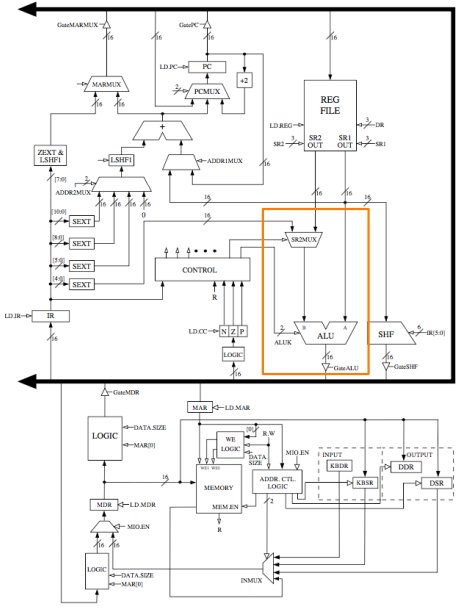
LC-3b Datapath



Driver `PC`

- `int inputOfGatePC;`
- Provided from PC

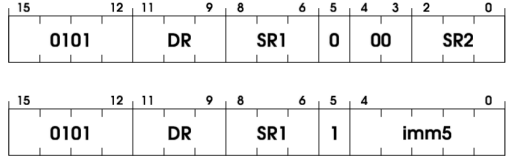
LC-3b Datapath



Driver `ALU`

- `int inputOfGateALU`
- `SR2MUX` is controlled by `IR[5:5]` (refer to `AND` instruction)

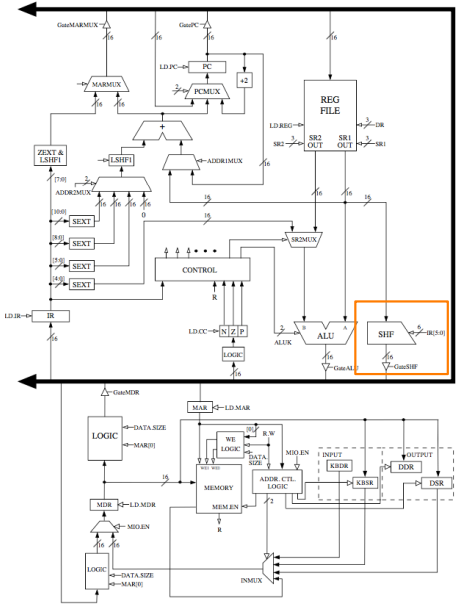
Encodings



Operation

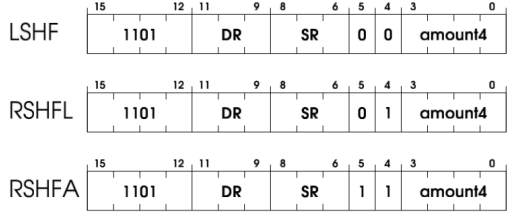
```
if (bit[5] == 0)
  DR = SR1 AND SR2;
else
  DR = SR1 AND SEXT(imm5);
setcc();
```

LC-3b Datapath



Driver `SHF`

- `int inputOfGateSHF`

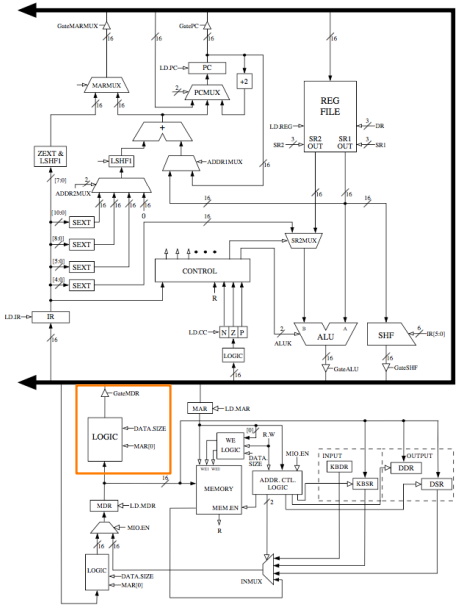


Operation

```
if (bit[4] == 0)
  DR = LSHF(SR, amount4);
else
  if (bit[5] == 0)
    DR = RSHF(SR, amount4, 0);
  else
    DR = RSHF(SR, amount4, SR[15]);
setcc();
```

- Implementation has been provided

LC-3b Datapath



Driver `MDR`

- `DATA_SIZE`: if 0 then byte base; if 1 then word base.
- If byte base, should check `MAR[0]`
  - If 0: `MDR[7:0]`
  - If 1: `MDR[15:8]`

Help on GateALU

- Input 1: SR1 (please refer to Figure C.6 (b))
- Input 2: SR2MUX, select the second source operand (see instructions AND XOR NOT)

```
/* input of GateALU */
inputOfGateALU = blockALU
(
    GetALUK(CURRENT_LATCHES.MICROINSTRUCTION),
    CURRENT_LATCHES.REGS[blockSR1MUX(GetSR1MUX(CURRENT_LATCHES.
       MICROINSTRUCTION), CURRENT_LATCHES.IR)],
    blockSR2MUX(xxx)
);
```

Overview

Introduction

Lab3-2 Assignment

Golden Results

Lab3-2 Assignment

```
515
516 /*
517  * Datapath routine emulating operations before driving the bus.
518  * Evaluate the input of tristate drivers
519  *       Gate_MARMUX,
520  *       Gate_PC,
521  *       Gate_ALU,
522  *       Gate_SHF,
523  *       Gate_MDR.
524  */
525 void eval_bus_drivers()
526 {
527     /*
528      *  Lab3-2 assignment
529      *
530      */
531 }
532
533
534 /*
535  * Datapath routine for driving the bus from one of the 5 possible
536  * tristate drivers.
537  */
538 void drive_bus()
539 {
540     /*
541      *  Lab3-2 assignment
542      *
543      */
544 }
545
```

Overview

Introduction

Lab3-2 Assignment

Golden Results

Assignment Package

- `lc3bsim3-2.c, lc3bsim3-2.h`: codes to work on
- `libems3-2.a`: library
- `ucode3`: FSM
- `Makefile`
- `bench`: folder with benchmarks

Run the simulator:

1. `make`, then binary "lc3bsim3-2" is generated
2. `./lc3bsim3-2 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
```

Golden Results – case `count10.cod`

1. run 7

```
Simulating for 7 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
MemCycleCnt = 1
MEM_EN = 0, R_W = 0, WE0 = 0, WE1 = 0
```

Golden Results – case `count10.cod` (cont.)

2. rdump

```
Current register/bus values :
--------------------------------------
Cycle Count  : 7
PC           : 0x3002
IR           : 0xe005
STATE_NUMBER : 0x0020

BUS          : 0xe005
MDR          : 0xe005
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 `count10.cod` (cont.)

3. Go on run 2

```
Simulating for 2 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
```

Golden Results – case `count10.cod` (cont.)

4. rdump

```
Current register/bus values :
--------------------------------------
Cycle Count  : 9
PC           : 0x3002
IR           : 0xe005
STATE_NUMBER : 0x0012

BUS          : 0x300c
MDR          : 0xe005
MAR          : 0x3000
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x300c
1: 0x0000
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```

Golden Results – case `count10.cod` (cont.)

5. Go on run 7

```
Simulating for 7 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
MemCycleCnt = 1
MEM_EN = 0, R_W = 0, WE0 = 0, WE1 = 0
```

Golden Results – case `count10.cod` (cont.)

6. rdump

```
Current register/bus values :
--------------------------------------
Cycle Count  : 16
PC           : 0x3004
IR           : 0x6200
STATE_NUMBER : 0x0020

BUS          : 0x6200
MDR          : 0x6200
MAR          : 0x3002
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x300c
1: 0x0000
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```

Golden Results – case `count10.cod` (cont.)

7. Go on run 7

```
Simulating for 7 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 = 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 `count10.cod` (cont.)

8. rdump

```
Current register/bus values :
--------------------------------------
Cycle Count  : 23
PC           : 0x3004
IR           : 0x6200
STATE_NUMBER : 0x001b

BUS          : 0x0000
MDR          : 0x000a
MAR          : 0x300c
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x300c
1: 0x0000
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```

Golden Results – case `count10.cod` (cont.)

9. Go on run 1

```
Simulating for 1 cycles...

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

Golden Results – case `count10.cod` (cont.)

10. rdump

```
Current register/bus values :
--------------------------------------
Cycle Count  : 24
PC           : 0x3004
IR           : 0x6200
STATE_NUMBER : 0x0012

BUS          : 0x000a
MDR          : 0x000a
MAR          : 0x300c
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x300c
1: 0x000a
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```

Golden Results – case `count10.cod` (cont.)

11. Go on run 1

```
Simulating for 1 cycles...

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

Golden Results – case `count10.cod` (cont.)

12. rdump

```
Current register/bus values :
--------------------------------------
Cycle Count  : 25
PC           : 0x3006
IR           : 0x6200
STATE_NUMBER : 0x0021

BUS          : 0x3004
MDR          : 0x000a
MAR          : 0x3004
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x300c
1: 0x000a
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```

Golden Results – case `count10.cod` (cont.)

13. Go on run 1

```
Simulating for 1 cycles...

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

Golden Results – case `count10.cod` (cont.)

14. rdump

```
Current register/bus values :
--------------------------------------
Cycle Count  : 26
PC           : 0x3006
IR           : 0x6200
STATE_NUMBER : 0x0021

BUS          : 0x0000
MDR          : 0x0000
MAR          : 0x3004
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x300c
1: 0x000a
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```

Golden Results – case `count10.cod` (cont.)

15. Go on run 4

```
Simulating for 4 cycles...

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 `count10.cod` (cont.)

16. rdump

```
Current register/bus values :
--------------------------------------
Cycle Count  : 30
PC           : 0x3006
IR           : 0x6200
STATE_NUMBER : 0x0023

BUS          : 0x0000
MDR          : 0x127f
MAR          : 0x3004
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x300c
1: 0x000a
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```

Golden Results – case `count10.cod` (cont.)

17. Go on run 3

```
Simulating for 3 cycles...

MemCycleCnt = 1
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
```

Golden Results – case `count10.cod` (cont.)

18. rdump

```
Current register/bus values :
--------------------------------------
Cycle Count  : 33
PC           : 0x3006
IR           : 0x127f
STATE_NUMBER : 0x0012

BUS          : 0x0009
MDR          : 0x127f
MAR          : 0x3004
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x300c
1: 0x0009
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