

# CENG3420

## Lab 2-2: LC-3b Simulator

**Bei Yu**

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

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

Mar. 06, 2017



香港中文大學  
The Chinese University of Hong Kong

# Overview

Basis

LC-3b Example: Count From 10 To 1

Tasks



# Overview

Basis

LC-3b Example: Count From 10 To 1

Tasks



# The Slides are self-contained? **NO!**

Do please refer to following two documents:

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



# Notations

DR

- ▶ Destination register

LSHF (A, b)

- ▶ Shift A to the **left** by b bits
- ▶ If A = 1111 1111 1111 1111, b = 5
- ▶ Then LSHF (A, b) = 1111 1111 1110 0000

MEM[addr]

- ▶ **Word** starting at the given memory address

setcc ()

- ▶ Set condition codes N, Z, P based on DR value

SEXT (A)

- ▶ **Sign-extend** A to 16 bits
- ▶ If A = 11 0000, SEXT (A) = 1111 1111 1111 0000



# Overview

Basis

LC-3b Example: Count From 10 To 1

Tasks



# LC-3b Example 2: Count from 10 to 1

count10.asm:

```
.ORIG x3000
LEA R0, TEN
LDW R1, R0, #0
START ADD R1, R1, #-1
      BRZ DONE
      BR START

DONE  TRAP x25
TEN   .FILL x000A
      .END
```

count10.cod:

```
0x3000
0xE005
0x6200
0x127F
0x0401
0x0FFD

0xF025
0x000A
```



# LEA: Load Effective Address

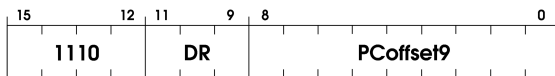
```
.ORIG x3000
LEA R0, TEN
LDW R1, R0, #0
START ADD R1, R1, #-1
BRZ DONE
BR START

DONE TRAP x25
TEN .FILL x000A
.END
```

```
0x3000
0xE005
0x6200
0x127F
0x0401
0x0FFD

0xF025
0x000A
```

► 0xE005 → 1110 000 000000101



1.  $DR = PC + 2 + \text{LSHF}(\text{SEXT}(\text{PCOffset9}), 1);$
2. `setcc();`





# LDW: Load Word

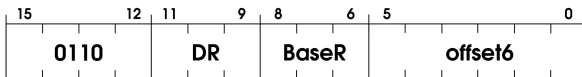
```
.ORIG x3000
LEA R0, TEN
LDW R1, R0, #0
START ADD R1, R1, #-1
      BRZ DONE
      BR START

DONE  TRAP x25
TEN   .FILL x000A
      .END
```

```
0x3000
0xE005
0x6200
0x127F
0x0401
0x0FFD

0xF025
0x000A
```

▶  $0x6200 \rightarrow 0110\ 001\ 000\ 0000000$



1.  $DR = MEM[BaseR + LSHF(SEXT(offset6), 1)]$ ;
2. `setcc()`;



# ADD: Addition

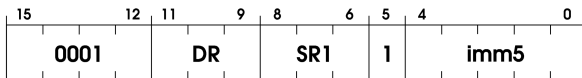
```
.ORIG x3000
LEA R0, TEN
LDW R1, R0, #0
START ADD R1, R1, #-1
BRZ DONE
BR START

DONE TRAP x25
TEN .FILL x000A
.END
```

```
0x3000
0xE005
0x6200
0x127F
0x0401
0x0FFD

0xF025
0x000A
```

▶ 0x127F → 0001 001 001 1 11111



1. DR = SR1 + SEXT(imm5);
2. setcc();



# Sample: Codes of Addition

```
461 {
462     case 1: /* add, and */
463     case 5:
464         DR = partVal(curInstr, 11, 9);
465         SR1 = partVal(curInstr, 8, 6);
466         if (partVal(curInstr, 5, 5)) /* imm5 */
467         {
468             imm5 = partVal(curInstr, 4, 0);
469             value = SEXT(imm5, 5);
470         }
471         else
472         {
473             SR2 = partVal(curInstr, 2, 0);
474             value = CURRENT_LATCHES.REGS[SR2];
475         }
476         if (opCode == 1)
477         {
478             NEXT_LATCHES.REGS[DR] = Low16bits(CURRENT_LATCHES.REGS[SR1] + value);
479         }
480         else if (opCode == 5)
481         {
482             NEXT_LATCHES.REGS[DR] = Low16bits(CURRENT_LATCHES.REGS[SR1] & value);
483         }
484         setCC(NEXT_LATCHES.REGS[DR]);
485         break;
486     }
```

1. `DR = SR1 + SEXT(imm5);`
2. `setcc();`



# BR: Conditional Branch

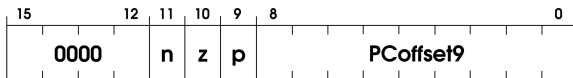
```
.ORIG x3000
LEA R0, TEN
LDW R1, R0, #0
START ADD R1, R1, #-1
      BRZ DONE
      BR START

DONE  TRAP x25
TEN   .FILL x000A
      .END
```

```
0x3000
0xE005
0x6200
0x127F
0x0401
0x0FFD

0xF025
0x000A
```

▶ 0x0401 → 0000 010 000000001



1. if (CURRENT\_LATCHES.Z) then:
2. PC = PC + 2 + LSHF(SEXT(PCoffset9), 1);



# BR: Conditional Branch

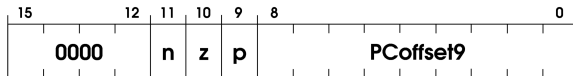
```
.ORIG x3000
LEA R0, TEN
LDW R1, R0, #0
START ADD R1, R1, #-1
      BRZ DONE
      BR START

DONE  TRAP x25
TEN   .FILL x000A
      .END
```

```
0x3000
0xE005
0x6200
0x127F
0x0401
0x0FFD

0xF025
0x000A
```

► 0x0401 → 0000 111 111111101



1. if (CURRENT\_LATCHES.N || CURRENT\_LATCHES.Z || CURRENT\_LATCHES.P) then:
2. PC = PC + 2 + LSHF(SEXT(PCoffset9), 1);



# TRAP x25: Halt

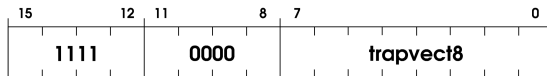
```
.ORIG x3000
LEA R0, TEN
LDW R1, R0, #0
START ADD R1, R1, #-1
      BRZ DONE
      BR START

DONE  TRAP x25
TEN   .FILL x000A
      .END
```

```
0x3000
0xE005
0x6200
0x127F
0x0401
0x0FFD

0xF025
0x000A
```

► 0x0401 → 1111 0000 00100101



1.  $R7 = PC + 2;$
2.  $PC = MEM[LSHF(ZEXT(trapvect8), 1)];$



# Sample: Codes of TRAP x25

```
523     case 15: /* TRAP */
524         trapvect8 = partVal(curInstr, 7, 0);
525         NEXT_LATCHES.REGS[7] = Low16bits(CURRENT_LATCHES.PC + 2);
526         NEXT_LATCHES.PC = memWord(trapvect8<<1);
527         break;
528
```



1.  $R7 = PC + 2;$
2.  $PC = MEM[LSHF(ZEXT(trapvect8), 1)];$



# Overview

Basis

LC-3b Example: Count From 10 To 1

Tasks





## Task 2: partVal () function

- ▶ Implement `int partVal (int, int, int);`
- ▶ Then TRAP instruction is completed;

```
407
408 /* return the corresponding value of instr[hBit:lBit] */
409 int partVal (int instr, int hBit, int lBit)
410 {
411     /*
412      * Lab2-2 assignment.
413      */
414     return 0;
415 }
416
```



## Task 2: SEXT () & setCC () functions

- ▶ Implement `int SEXT (int , int)`
- ▶ Implement `void setCC(int)`
- ▶ Then ADD instruction is completed

```
417 /* sign extend the imm to 16 bits
418 * 'width' is the bit width before imm is extended*/
419 int SEXT (int imm, int width)
420 {
421     /*
422     * Lab2-2 assignment.
423     */
424     return 0;
425 }
426
427 /* set condition code */
428 void setCC(int num)
429 {
430     /*
431     * Lab2-2 assignment.
432     */
433 }
```



# Golden Result of Task 2: bench/testTask2-2.cod

## 1. run 1

Instructions:

```
process_instruction() | curInstr = 0x1261
```

Registers:

```
Instruction Count : 1
PC                : 0x3002
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x0000
1: 0x0001
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```



# Golden Result of Task 2: bench/testTask2-2.cod

## 2. Go on **run 1**

Instructions:

```
process_instruction() | curInstr = 0x1261
```

Registers:

```
Instruction Count : 2  
PC                : 0x3004  
CCs: N = 0  Z = 0  P = 1  
Registers:  
0: 0x0000  
1: 0x0002  
2: 0x0000  
3: 0x0000  
4: 0x0000  
5: 0x0000  
6: 0x0000  
7: 0x0000
```



# Task 3: Parse LEA, LDW, BR instructions

- ▶ Finish the following parts

```
486  
487     case 0: /* br */  
488         /* Lab2-2 assignment: */  
489         break;  
490
```

```
501  
502     case 6: /* ldw */  
503         /* Lab2-2 assignment: */  
504         break;  
505
```

```
517  
518     case 14: /* lea */  
519         /* Lab2-2 assignment: */  
520         break;  
521
```

- ▶ Please refer implementations of ADD, TRAP
- ▶ Then the simulator can work on `count10.cod`



# Golden Result of Task 3: bench/count10.cod

## 1. run 2

Instructions:

```
process_instruction() | curInstr = 0xe005  
process_instruction() | curInstr = 0x6200
```

Registers:

```
Instruction Count : 2  
PC                : 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 Result of Task 3: bench/count10.cod

## 2. Go on run 6

### Instructions:

```
process_instruction() | curInstr = 0x127f
process_instruction() | curInstr = 0x0401
process_instruction() | curInstr = 0x0ffd
process_instruction() | curInstr = 0x127f
process_instruction() | curInstr = 0x0401
process_instruction() | curInstr = 0x0ffd
```

### Registers:

```
Instruction Count : 8
PC                : 0x3004
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x300c
1: 0x0008
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```



# Golden Result of Task 3: bench/count10.cod

## 3. Go on run 12

Instructions:

```
process_instruction() | curInstr = 0x127f
process_instruction() | curInstr = 0x0401
process_instruction() | curInstr = 0x0ffd
process_instruction() | curInstr = 0x127f
process_instruction() | curInstr = 0x0401
process_instruction() | curInstr = 0x0ffd
process_instruction() | curInstr = 0x127f
process_instruction() | curInstr = 0x0401
process_instruction() | curInstr = 0x0ffd
process_instruction() | curInstr = 0x127f
process_instruction() | curInstr = 0x0401
process_instruction() | curInstr = 0x0ffd
```

Registers:

```
Instruction Count : 20
PC                 : 0x3004
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x300c
1: 0x0004
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```





# Golden Result of Task 3: bench/count10.cod

## 4. Go on run 12

Instructions:

```
process_instruction() | curInstr = 0x127f
process_instruction() | curInstr = 0x0401
process_instruction() | curInstr = 0x0ffd
process_instruction() | curInstr = 0x127f
process_instruction() | curInstr = 0x0401
process_instruction() | curInstr = 0x0ffd
process_instruction() | curInstr = 0x127f
process_instruction() | curInstr = 0x0401
process_instruction() | curInstr = 0x0ffd
process_instruction() | curInstr = 0x127f
process_instruction() | curInstr = 0x0401
process_instruction() | curInstr = 0x0ffd
process_instruction() | curInstr = 0x127f
process_instruction() | curInstr = 0x0401
process_instruction() | curInstr = 0xf025
```

Registers:

```
Instruction Count : 32
PC                 : 0x0000
CCs: N = 0  Z = 1  P = 0
Registers:
0: 0x300c
1: 0x0000
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x300c
```

