Programming Studio #3

ECE 190

Programming Studio #3

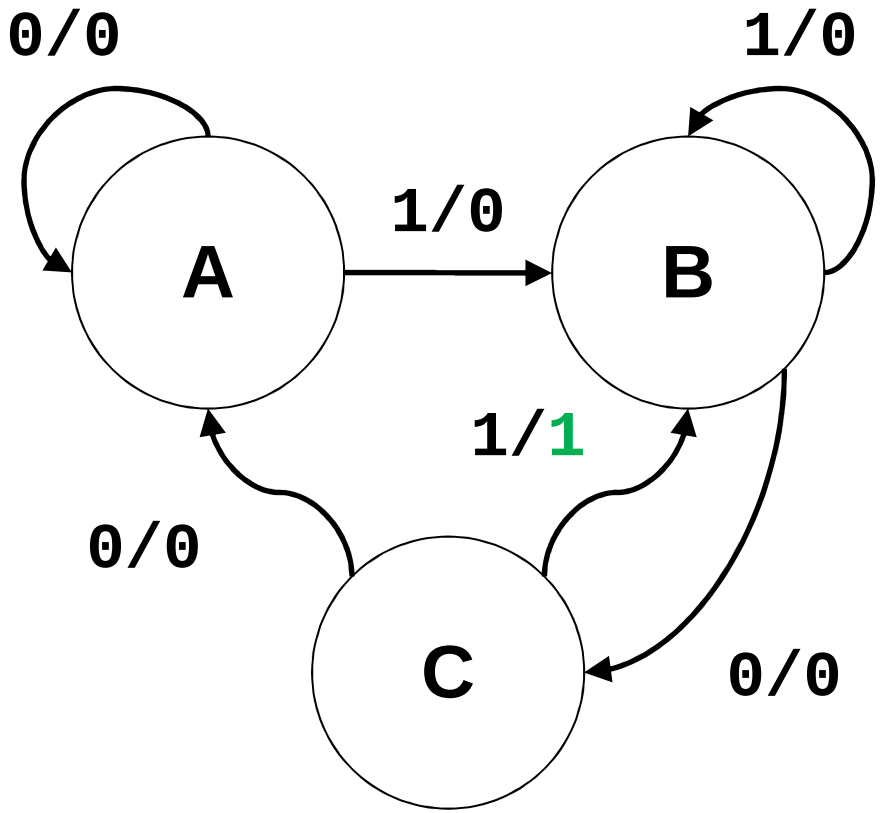
- Concepts this week:
  - Finite state machines
  - LC-3 ISA
  - Machine language
  - LC-3 simulator

Announcements

- MP1 due Weds. 2/17 at 5:00pm
  - Read handout carefully to not lose points
  - Office Hours increased starting next Weds.
- Exam 1: Thurs. 2/25, 7:00pm-9:00pm
  - Feb. 18th hard deadline for conflict requests
  - Practice exams on course website
  - Exam review TBA
- Look at the "Student Manual" on the course website

Finite state-machines (FSM)

- Sequential circuit representation (memory)
- Detecting **101** in a string of bits
  - Notation of state transition: **input/output**
  - Input: **110100001010**1…
  - Output: **000100000010**1…
  - States:
    - A (start / reset [none of pattern to be detected seen])
    - B (1 seen in input)
    - C (10 seen in input)



LC-3 Overview

- What is…
  - the LC-3?
    - What are its address space and addressability?
      - **2^16 addresses** and **16 bits / address**
    - What are registers? How many are there? **8**
    - What is the **PC**? **MAR**? **MDR**?
    - What is an instruction?
      - **Opcode** and **operands**
  - machine language?: `0001 010 000 0 00 001`
  - assembly?:

```
ADD  R2, R0,      R1
ADD  DR, SR1,     SR2
```

  - the result? `DR <- SR1 + SR2`
- How many opcodes are there?

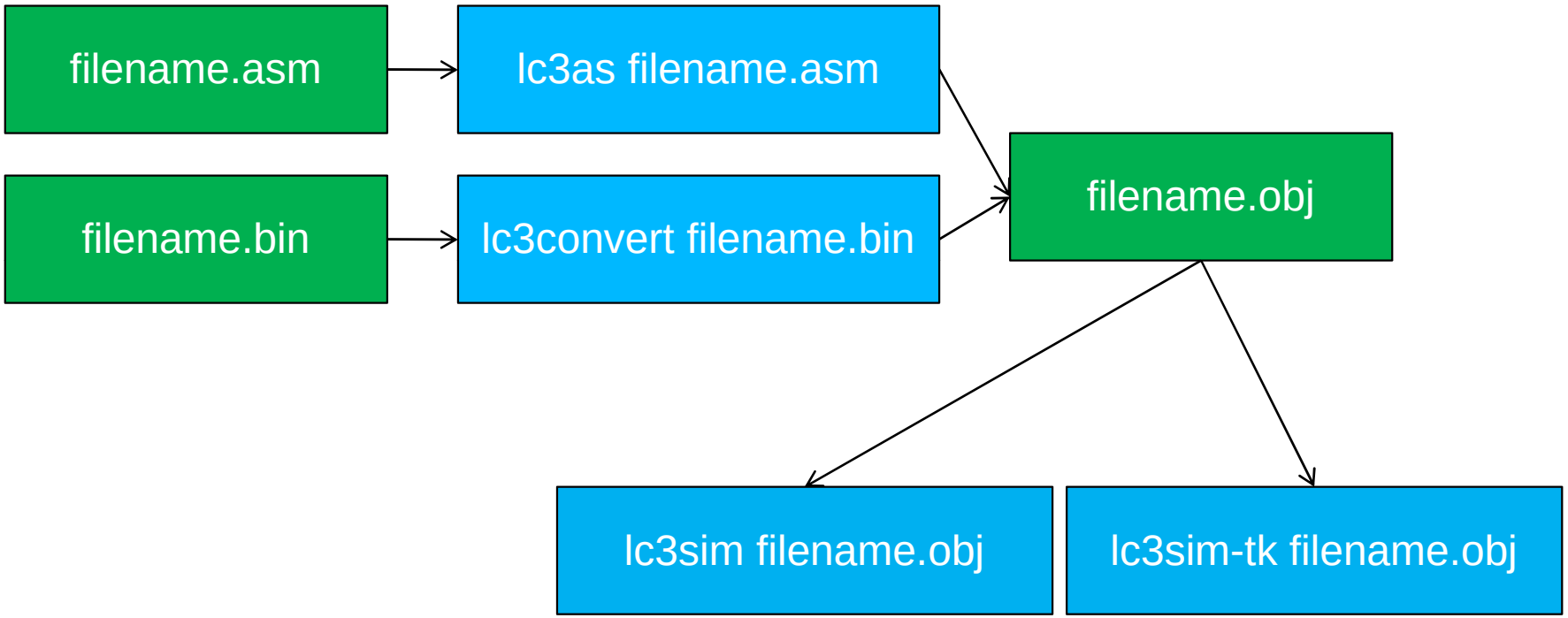
LC-3 Overview (cont)

- What are…
  - Condition codes?
    - single-bit registers modified on some instructions to indicate whether the result of the instruction was **negative** (N), **zero** (Z), or **positive** (P)
  - Addressing modes?
    - Immediate (literal): from the instruction
      - E.g., `ADD R2, R3, #7  ; R2 <- R3 + 7`
    - Register: from a register
      - E.g., `ADD R2, R3, R4  ; R2 <- R3 + R4`
    - Note: value in memory at address **xDEAD** is **M[xDEAD]** (sometimes **mem[xDEAD]**)
    - **PC'** is incremented **PC**, so **PC'=PC + 1**
    - PC-relative (LD, ST): **M[PC' + relative offset]**
    - Others for later: indirect and base+offset

LC-3 Tools

- **lc3convert**: converts human readable **machine code** files to LC-3 simulator readable files (**object** files)
  - Syntax: lc3convert filename.bin
- **lc3as**: converts human readable **assembly** into object files
  - Syntax: lc3as filename.asm
- **lc3sim**: command line LC-3 simulator
  - Syntax: lc3sim filename.obj
- **lc3sim-tk**: graphical LC-3 simulator
  - Syntax: lc3sim-tk filename.obj

LC-3 Programming Flowchart



Simulating a program

- Login, type the following:
  - ece190
  - lynx http://courses.ece.illinois.edu/ece190/discussion/spring10/ps03/mult.bin
- You are now in the lynx web browser, type the following:
  - `d, <down arrow>, <enter>, <enter>, q, y`
- You are now back on the command shell, type:
  - `lc3convert mult.bin`
  - `lc3sim mult.obj`
- You are now in the LC-3 simulator, type the following:
  - `help`
  - `list x3000`
  - `memory x3100  5`
  - `memory x3101  10`
  - `continue`
  - `list x3100`
- What is the value at memory address x3102?

mult.bin

```
0011 0000 0000 0000    ;                        start of code at x3000
0101 010 010 1 00000   ;x3000 AND R2,R2,#0      initialize: R2 <- 0
0010 100 0 1111 1110   ;x3001 LD  R4, xFE       load first value: R4 <- M[x3100]
0010 101 0 1111 1110   ;x3002 LD  R5, xFE       load second value: R5 <- M[x3101]
0001 010 010 0 00 100  ;x3003 ADD R2,R2,R4      loop: running sum: R2 <- R2 + R4
0001 101 101 1 11111   ;x3004 ADD R5,R5,#-1     decrement loop counter: R5 <- R5-1
0000 001 1 1111 1101   ;x3005 BRp x3003         continue looping if R5 is positive
0011 010 0 1111 1011   ;x3006 ST  R2, xFB       store result: M[x3102] <- R2
1111 0000 0010 0101    ;x3007 HALT              halt the lc-3
```

- First line states where program starts in memory (address x3000)
- Binary code on the left is separated into segments (**fields**)
  - Each instruction interprets bits differently
  - Look at the LC-3 Instruction Set on the course webpage (or in the book)
- Text on the right after semicolon are **comments**,
  - Give some meaning to what the code does
  - Required that you document and comment your code
- What does the program do?
- More details: see the **tutorial.pdf** file on the website and the **Student Manual**