Attendance Check-in Form: https://forms.gle/NwHr5xTxNuxotiuU8
Questions and Demos Sheet:
https://docs.google.com/spreadsheets/d/1gTwMj510B_CebOsuRi1V7-DXjQiavs2NAL7tBdcO-FQ/edit?usp=sharing

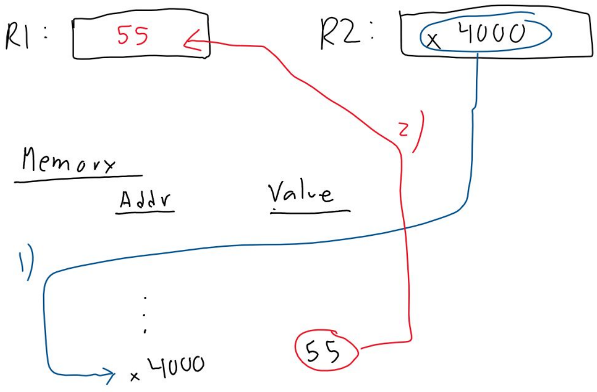
Outline

1. Recap: Lab2 and LC3 Assembly Basics
2. Points to highlight for today's lab

Recap: Lab2 and LC3 Assembly Basics

- Three memory addressing modes: Direct, Indirect, and Relative
    - Direct and Indirect both require labels to access memory, but Relative requires a *Base Register* holding a *memory address*.

LDR R1, R2, x0



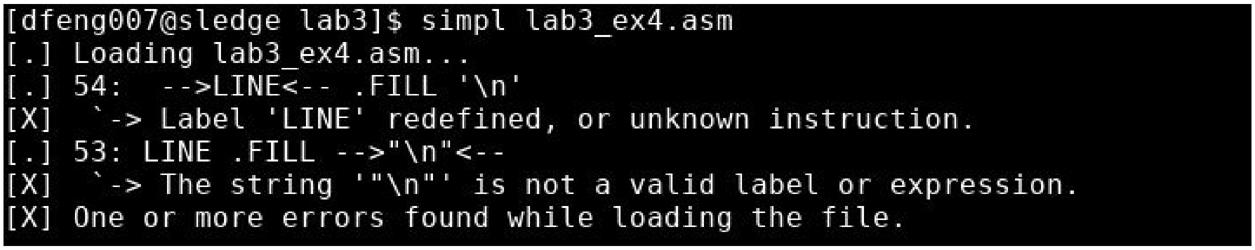
- Basics: Basic Skeleton of LC3 Assembly File
  In LC3, a basic program has the following format:

```
; <Header information>

.ORIG x3000         ; **ALL** your programs will start at x3000
; Instructions (i.e. LD, ST, ADD, AND, NOT, BR)
HALT                ; end of program code
; Local Data        ; pseudo-ops for hard-coding data go here

; Optional: Remote Data   ; More pseudo-ops go here, usually after
                          ; another .ORIG declaration (e.g .ORIG x4000)

.END                ; .end is like the “}” after main() in C++.
                          ; It means “no more code or data”
```

- Basics: Using Linux, running processes in the background
- Review: Reading feedback from Gradescope:
  https://piazza.com/class/kbjm3xq2sl9595?cid=8

Today’s Lab

- Working with arrays in assembly
    - We use .BLKW to reserve a certain amount of space for an array in memory
    - We will be storing characters from the user in the reserved spaces
- Key question: How can we traverse an array in assembly?
    - Which of the three addressing modes would be the best to do this? Which one of them is the most dynamic?
- Sentinel control loops
    - Instead of using a counter for a loop, we’re checking for a specific “sentinel value” to determine when to end the loop
    - How can we check for a specific character?
        - First think about how we do so in a higher-level language
        - Hint: (Highlight this line to see the hint)
    - Note that we have **TWO** different sentinel characters: one to tell us when to stop taking input from the user, and one to tell us to stop reading the array in memory
        - For the second sentinel character (for traversing the array for output), think about how PUTS works
- If your SIMPL window does not open, please check your terminal output, it references any syntax errors



```
[dfeng007@sledge lab3]$ simpl lab3_ex4.asm
[.] Loading lab3_ex4.asm...
[.] 54:  -->LINE<-- .FILL '\n'
[X]  `-> Label 'LINE' redefined, or unknown instruction.
[.] 53: LINE .FILL -->"\n"<--
[X]  `-> The string '"\n"' is not a valid label or expression.
[X] One or more errors found while loading the file.
```

- Single Quotes vs Double Quotes

```
; <Header information>

.ORIG x3000           ; **ALL** your programs will start at x3000
; Instructions (i.e. LD, ST, ADD, AND, NOT, BR)
HALT                  ; end of program code
; Local Data          ; pseudo-ops for hard-coding data go here

 CHAR .FILL 'a'
 STRING .STRINGZ "test"
```

Useful reference for basic I/O:
https://docs.google.com/document/d/14vaOKv7-IpuY8VWcoQnq8opI5CpcLHay7sTZG4BzoY4/edit