ECE 252: Intro to Computer Engineering

Week 12 Discussion

Attendance via TopHat

Course code: **265393**

Attendance code:

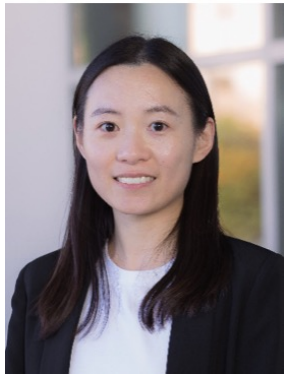
2017

ECE Discovery Panel Series

Get to know a technical area:
Applied Electromagnetics and Acoustics

Application Areas -- Advanced Electives -- Job Opportunities

Acoustic sensing, microwave ablation, electronic jamming, wireless power transfer



Tuesday, Nov 19 ,
12:20-12:50pm
1413 EH (Cheney Room)

Come for the insights, stay for the pizza!

Grab the latest ECE swag!

Subroutines

Course code: **265393**
Attendance code: **2017**

- Why have subroutines?
  - The same sequence of instructions needs to be used repeatedly in a program
    - Avoid copy/paste errors
    - Make it easier to modify/maintain
  - Create a library of functions that can be used by you and other programmers
    - Reusability!
  - And who wants to look at a massive program and try to figure out why it doesn't work?
    - Yuck!
- Subroutines support the development of **modular** software

Calling a Subroutine

Course code: **265393**
Attendance code: **2017**

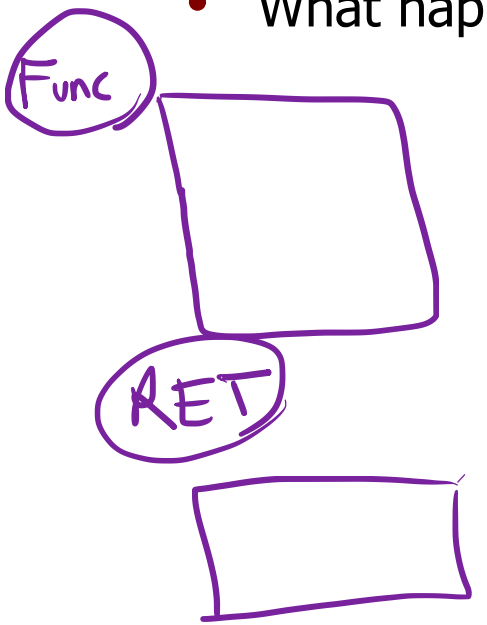
- **JSR label** ; **J**ump to **S**ub**R**outine
  - **R7** ← **PC**$^+$, **PC** ← **PC**$^+$ + SEXT(PCoffset11)
  - Range is approximately +/-1024 words

- What if the subroutine is further away?

- **JSRR BaseR** ; **J**ump to **S**ub**R**outine (**R**egister)
  - **temp** ← **PC**$^+$, **PC** ← **BaseR**, **R7** ← **temp**
  - **JSRR R7** will work as expected
    - In reality, we can update R7 and PC simultaneously and there is no need for a temporary register

Returning from a Subroutine

- To return from a subroutine, the subroutine code must use the value in **R7**
  - Execute the **RET** instruction, which does **PC** ← **R7**
  - Programming tip: Put the **RET** instruction at the end of the subroutine right away!
    - What happens if you forget **RET**?



Course code: **265393**
Attendance code: **2017**

Returning from a Subroutine

- All subroutine calls will modify **R7**!
  - If a subroutine calls a subroutine, it needs to save **R7**, call the subroutine, and then restore **R7**

Course code: **265393**
Attendance code: **2017**

Returning from a Subroutine

- What happens if subroutine **A** calls subroutine **B**, but **A** does not save and restore **R7**?

Course code: **265393**
Attendance code: **2017**

Well-Behaved Subroutines

- If a subroutine uses a register (and changes its value), it could cause problems for the caller

- There are two approaches to prevent subroutines from corrupting the caller's state
  - Have the caller save all registers in use so they will not be corrupted by the subroutine (**caller-save**)
  - Have the subroutine save all registers it uses, so the caller can assume no registers (other than **R7**) are corrupted by the subroutine (**callee-save**)

- Callee-save is most commonly used
  - In LC-3 (and others), caller must still save/restore **R7**

Subroutine Parameters and Results

- To send information to a subroutine, the caller can use registers or memory
  - The subroutine code must assume that the caller put the information where it was expected to be
    - The subroutine code must be well-documented as to parameter locations and constraints!
    - The subroutine CANNOT possibly know whether or not the caller put the correct parameters in

Course code: **265393**
Attendance code: **2017**

Subroutine Parameters and Results

- The results from a subroutine can be returned in registers or memory
  - Common error: If returning a result in a register, be sure that the register is NOT saved and restored by the subroutine (or you'll lose your result!)

Using Return Values

- If result is returned in memory, you need to read from memory to see the result
- If the result is returned in a register…
  - You don't have to do anything special before using that register in any instruction that explicitly uses a register
  - What about branches? Can you assume a return value set the condition codes?
    - What if the subroutine saves/restores registers?
    - What if the subroutine is changed later?
  - Good (safe) practice is to **explicitly** set condition codes based on the returned value

branch based on value returned in R0

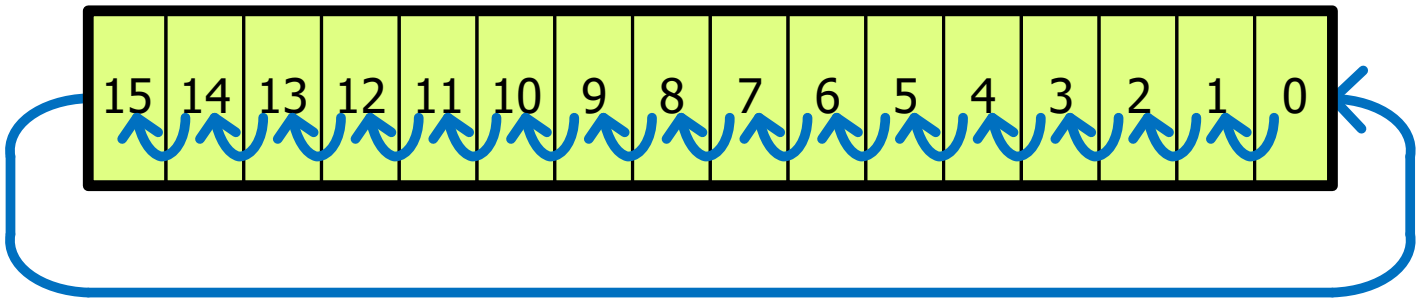
```
JSR mysub
ADD R0, R0, #0
BRz DoThing
```

How do you get good at this?

- **The only way to learn programming…**
  is to DO programming!
- The more you do, the easier it gets… because you know
  - …what the instructions do
  - …how to do stuff

Subroutine Practice 1

- Write a subroutine **ROL1** that rotates an operand left by one bit
  - Parameters: **R0** is value
  - Returns: **R0** is rotated result
- Subroutine should not corrupt any caller registers



  - Rotate left is a left shift except LSb←MSb instead of 0
    - Left shift (value) = 2(value) = (value) + (value)
    - Then, if original MSb was 1, add 1 to complete rotate

Rotate Left Examples

Rotate Left by 1

Before	0011 1100 0101 1010
After	0111 1000 1011 0100

Rotate Left by 1

Before	1010 0101 1100 0011
After	0100 1011 1000 0111

Rotate Left by 2

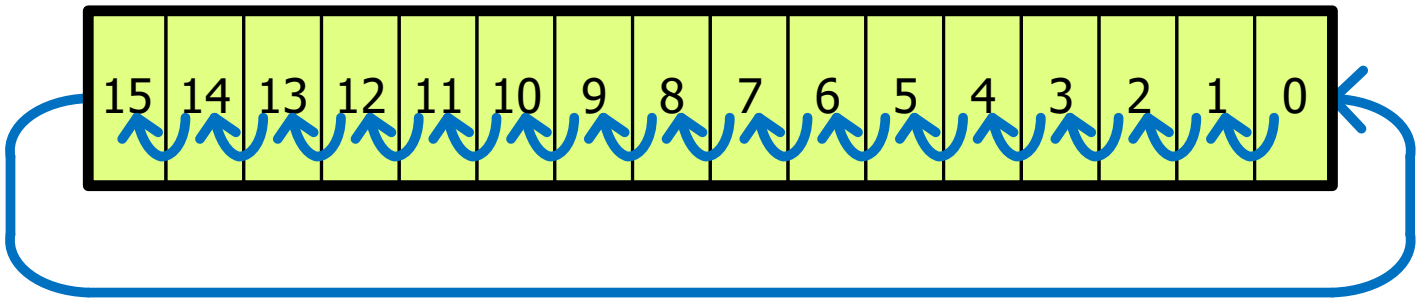
Before	0110 0010 1001 1111
After	1000 1010 0111 1101

Rotate Left by 4

Before	0110 0010 1001 1111
After	0010 1001 1111 0110

Subroutine Practice 1

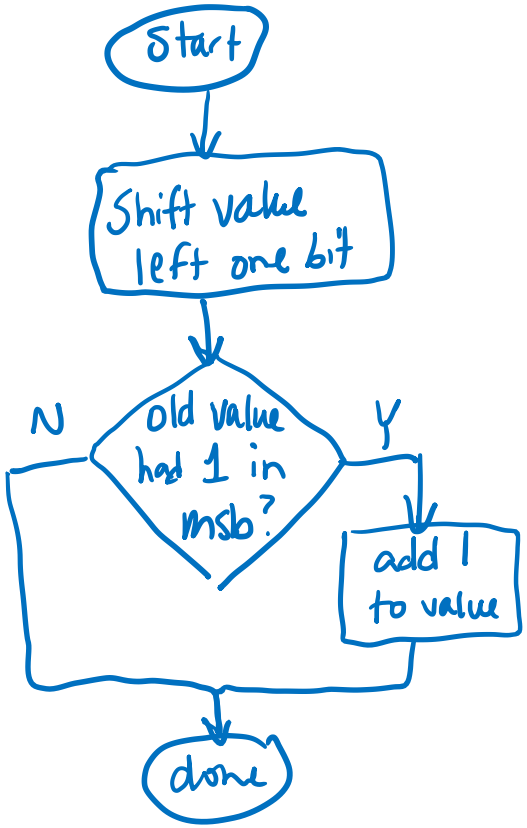
- Write a subroutine **ROL1** that rotates an operand left by one bit
  - Parameters: **R0** is value
  - Returns: **R0** is rotated result
- Subroutine should not corrupt any caller registers



  - Rotate left is a left shift except LSb←MSb instead of 0
    - Left shift (value) = 2(value) = (value) + (value)
    - Then, if original MSb was 1, add 1 to complete rotate

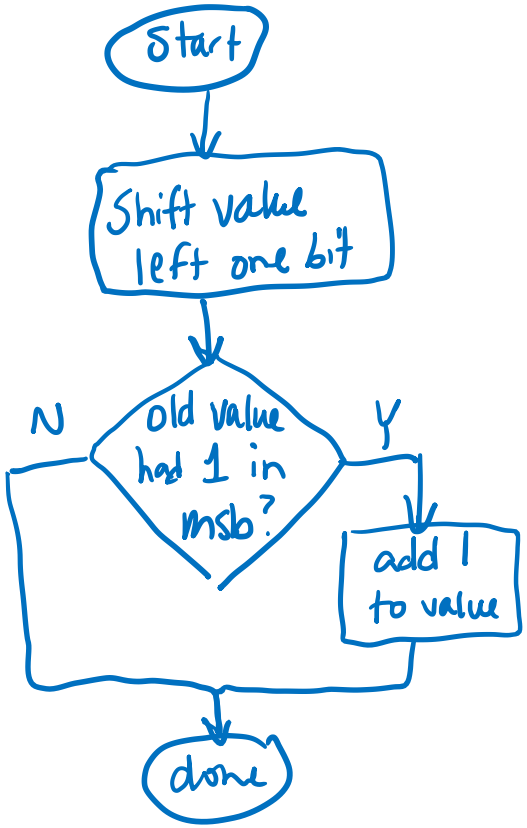
ROL1 rotates left by 1 bit

Parameters: **R0** is value Returns: **R0** is rotated result

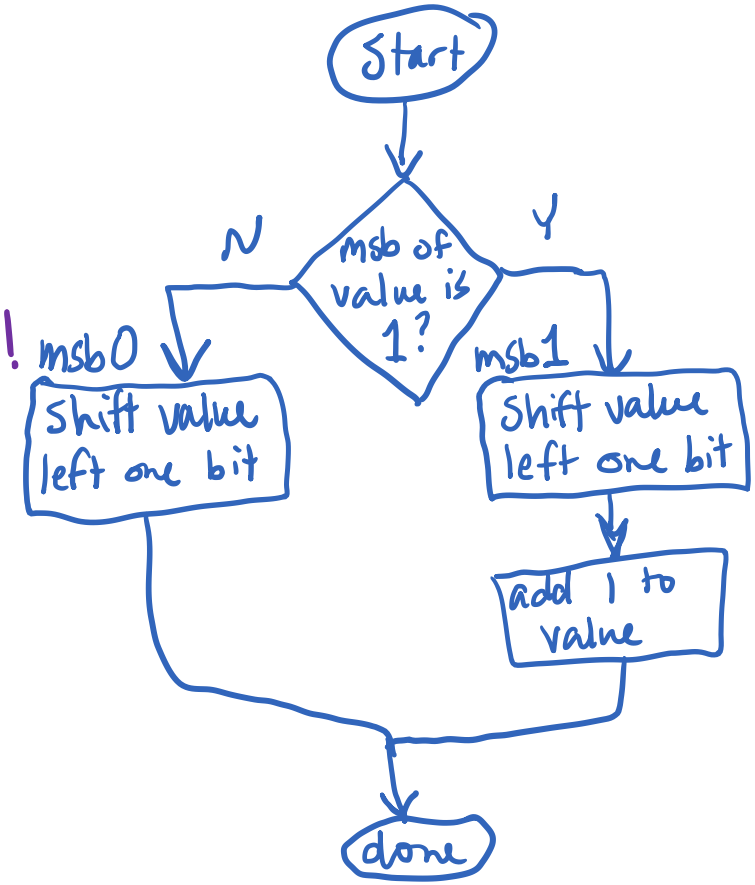


ROL1 rotates left by 1 bit

Parameters: **R0** is value Returns: **R0** is rotated result

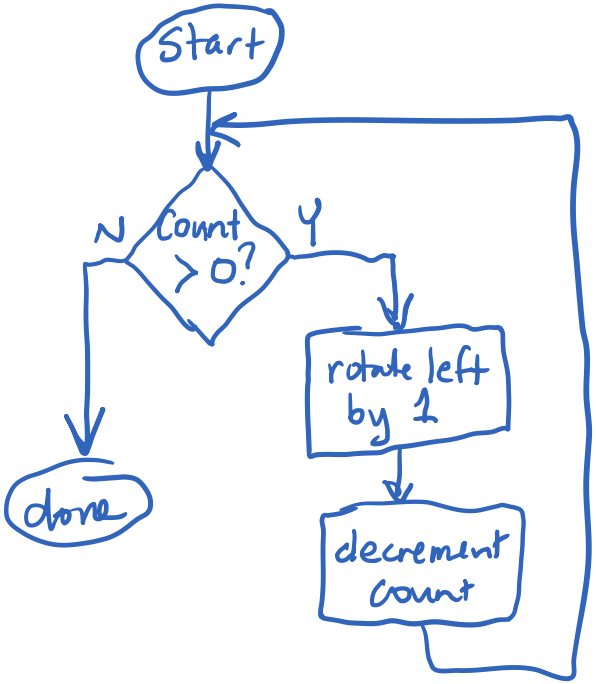


problem:
once we shift left, we dont know what the msb was!



ROL rotates left by N bits

Parameters: **R0** is value, **R1** is count (N) Returns: **R0** is rotated result



ROR rotates right by N bits

Parameters: **R0** is value, **R1** is count (N)

Returns: **R0** is rotated result

Wrapping Up

- Up Next:
  - I/O Concepts
  - LC-3 I/O
- Remember your videos and reading
  - Including the video quiz!
- Questions?