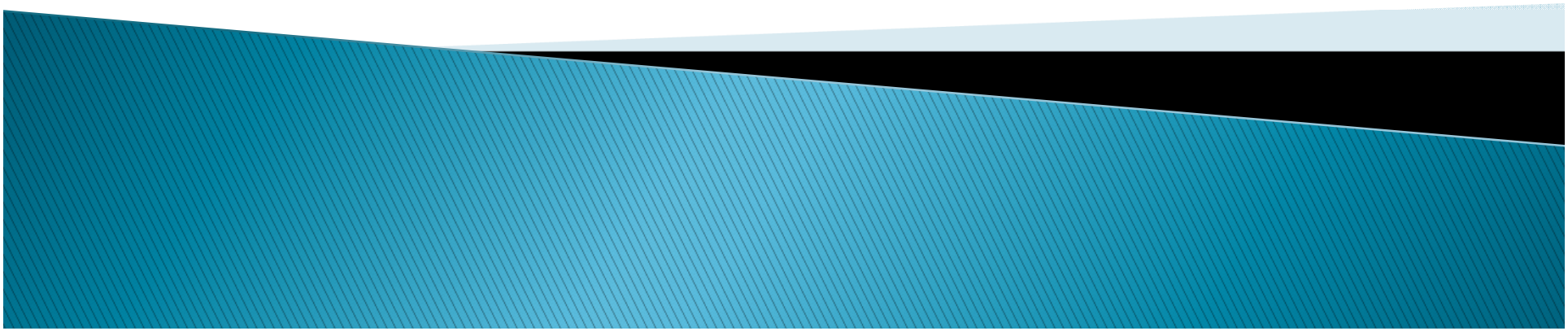
Computer Science 210 tutorial 6

LC-3 and Assembly code (3)

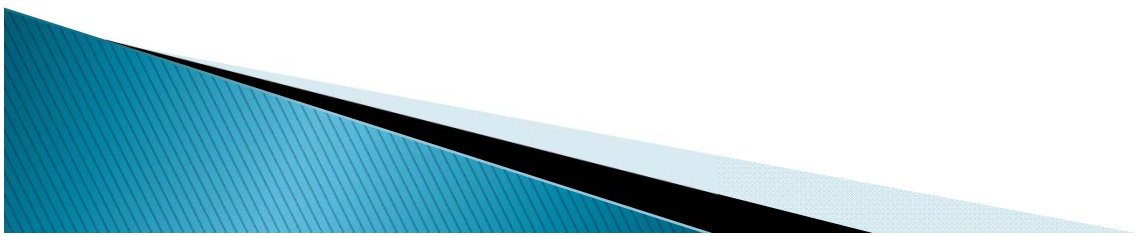


Tutorial 5 revision

- After tutorial 5, you have learnt
  - How to use some LC3 operations:
    - And, add, not, etc.
  - Load and store from/to memory
  - Inputs and outputs using:
    - GETC, IN, OUT, PUTS.
  - Start learning about BR (nzp)
  - Write a program to get user's name
- This tutorial will cover:
  - Answer to last exercise
  - Learn how to use Subroutine JSR
  - More exercises

Last tutorial examples/exercises

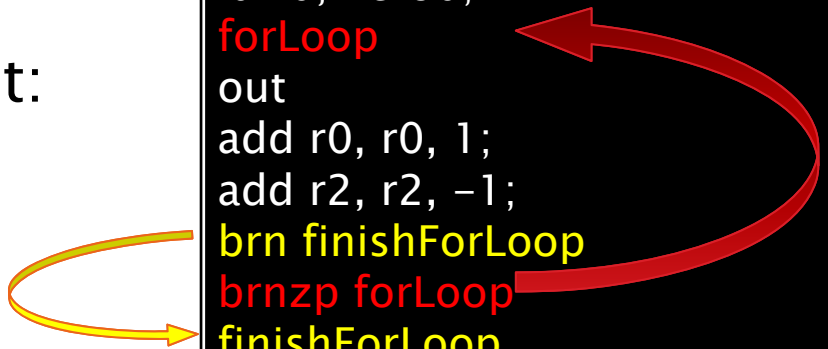
- Create an example to echo an user input, i.e:
  - Hi, what is your name?
  - David Beckham
  - Hi David Beckham, nice to meet you.
- Do exercises:
  - Input a number from 0 to 9
  - Print out all the number from 0 to that number
  - Example:
    - Input: 4
    - Output: 0 1 2 3 4



Exercise answer

- Steps need to complete:
  - Get input as a character
  - Turn that character to int by:
    - take away offset (‘0’)
    - N = ‘5’ – ‘0’
  - Make a for loop to print:
    - N times.
  - Start from
    - ‘0’
  - Use BR wisely

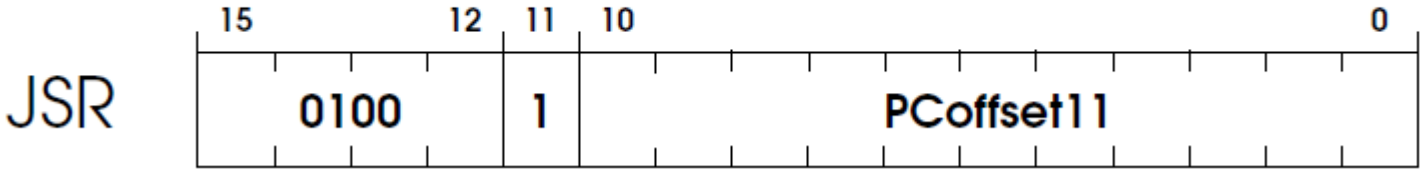
```
.orig x3000
ld r6, zero0
not r6, r6
add r6, r6, 1
lea r0, inputString
puts
getc
out
add r1, r0, 0
add r2, r1, r6
lea r0, outputString
puts
ld r0, zero0;
forLoop
out
add r0, r0, 1;
add r2, r2, -1;
brn finishForLoop
brnzp forLoop
finishForLoop
halt
inputString .stringz "Input: "
outputString .stringz "\nOutput: "
zero0 .fill 48
.end
```



Subroutine

JSR Label

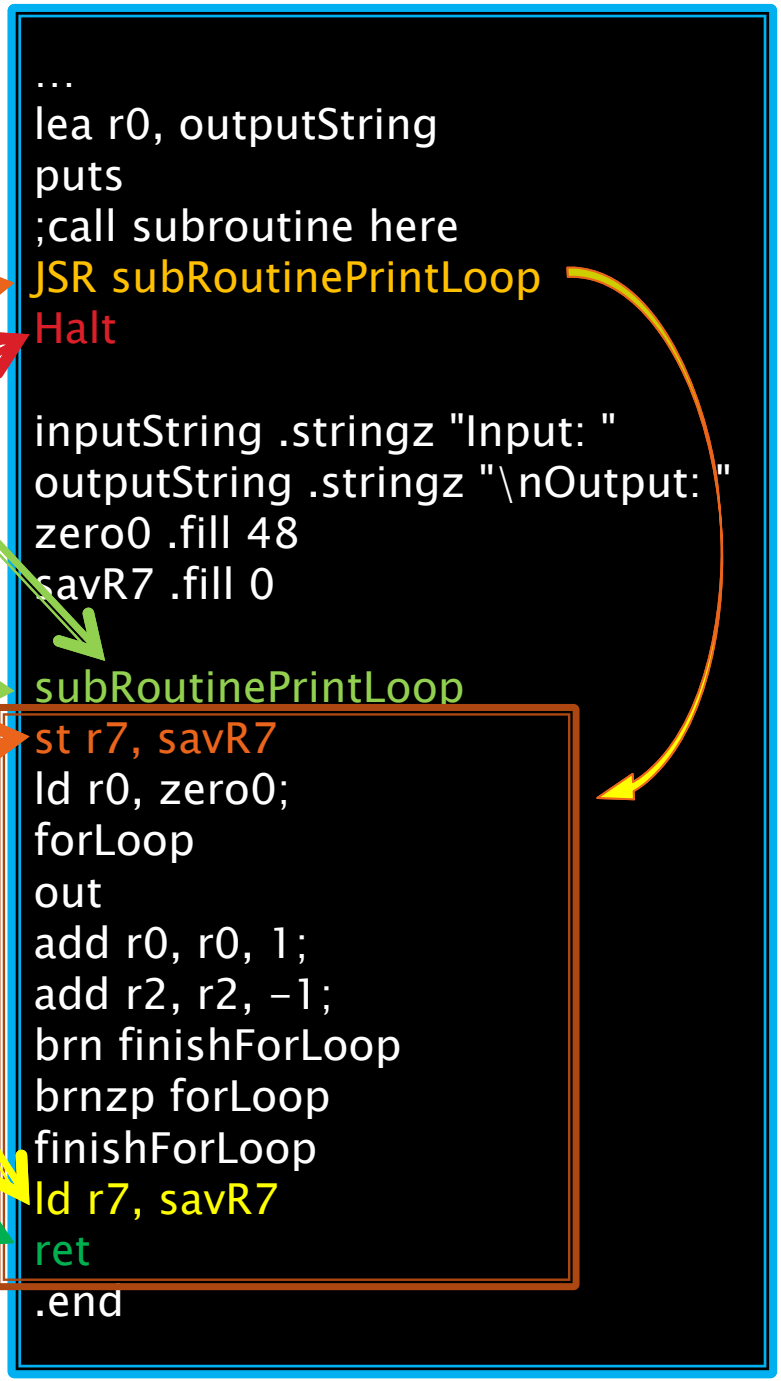
Encoding



- First, the incremented PC is saved in R7.
- This is the linkage back to the calling routine.
- Then the PC is loaded with the address of the first instruction of the subroutine, causing an unconditional jump to that address.
- Pc-offset has 11 bits: JSR can jump much further than BR (9 bits)

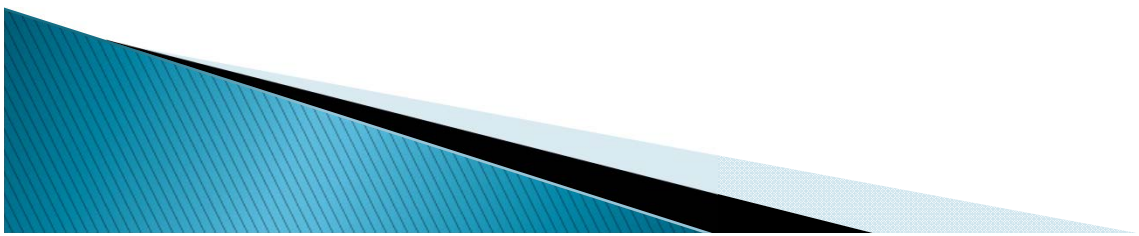
Code with JSR

- Subroutine has a label
- Call: JSR Label
  - R7 is updated with next address
  - PC is updated to subroutine
- In subroutine
  - should save R7
  - And load R7 when finish
- After subroutine finished, add operation RET
  - This puts R7 value to PC



Exercise 1

- Write a program/subroutine to check for ODD/EVEN number. Make it loops many times, finishes when user enter nothing:
  - *Please enter a number: 1234*
  - *Thanks, 1234 is an even number.*
  - *Please enter a number: 245*
  - *Thanks, 245 is an odd number.*
  - *Please enter a number:*
  - *Thanks, see you again.*
  - *---- halt----*



Exercise 2

- Write a parseInt, and toString subroutines
- parseInt:
  - Change character from R0 in to integer value
  - Store it back to R0
- toString:
  - Change integer value from R0 to character
  - Store it back to R0
- This will be useful in your assignment