**Computer Organization Fall 2024**

**HKBU-BNU United International College**

**Lab 8: LC3 Programming with Loops and Subroutines**

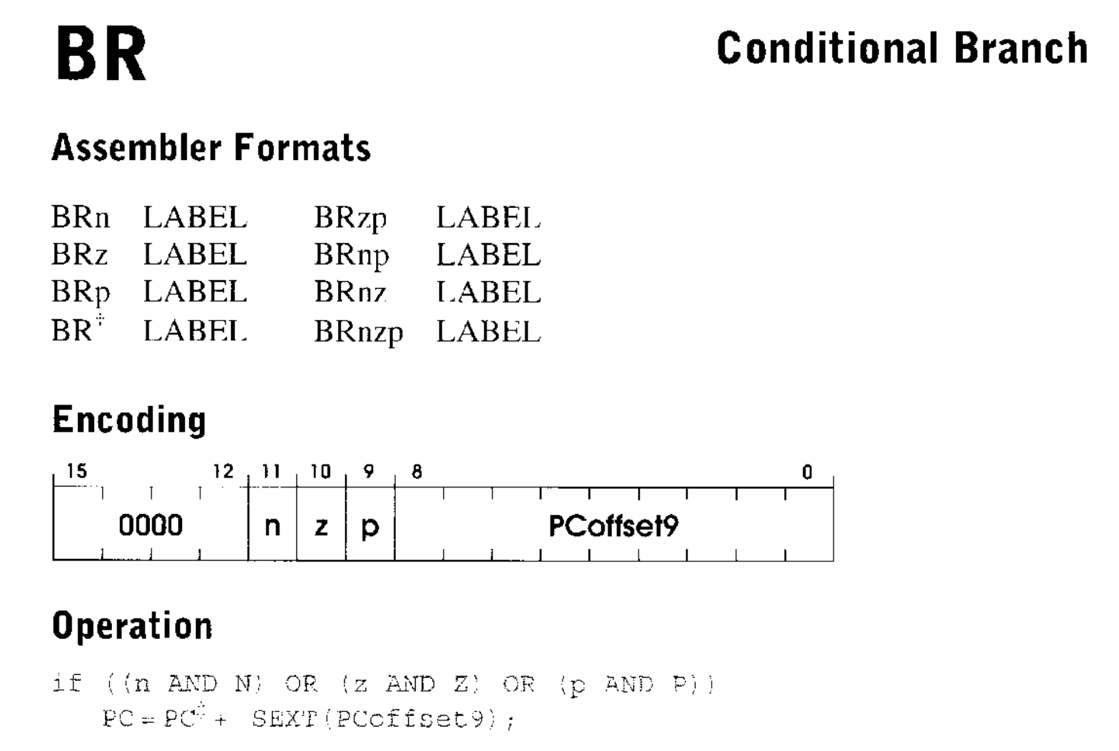
# Lab Objective

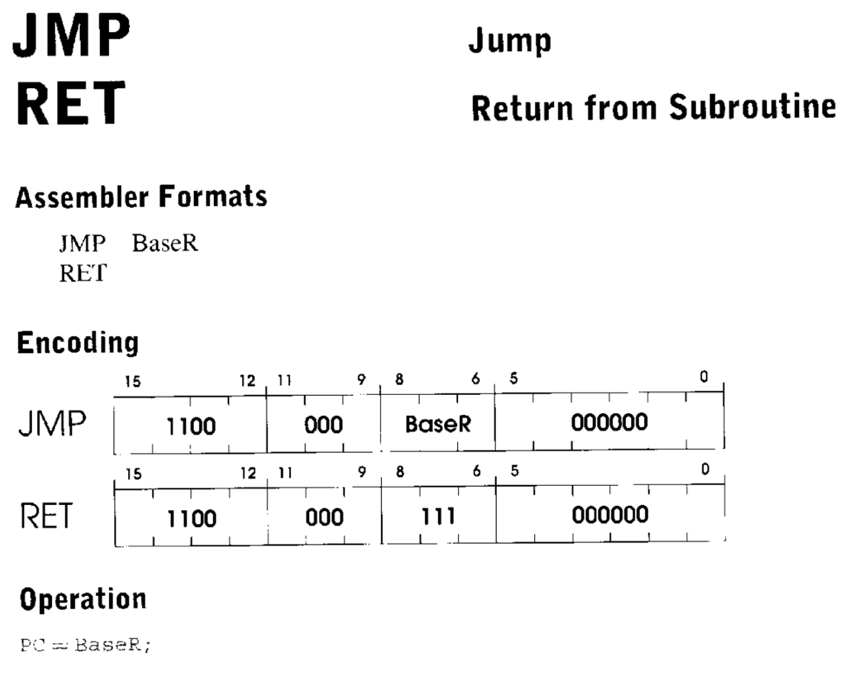
To learn how to write **loops** and **subroutines** using the LC-3 assembly language.

# Introduction

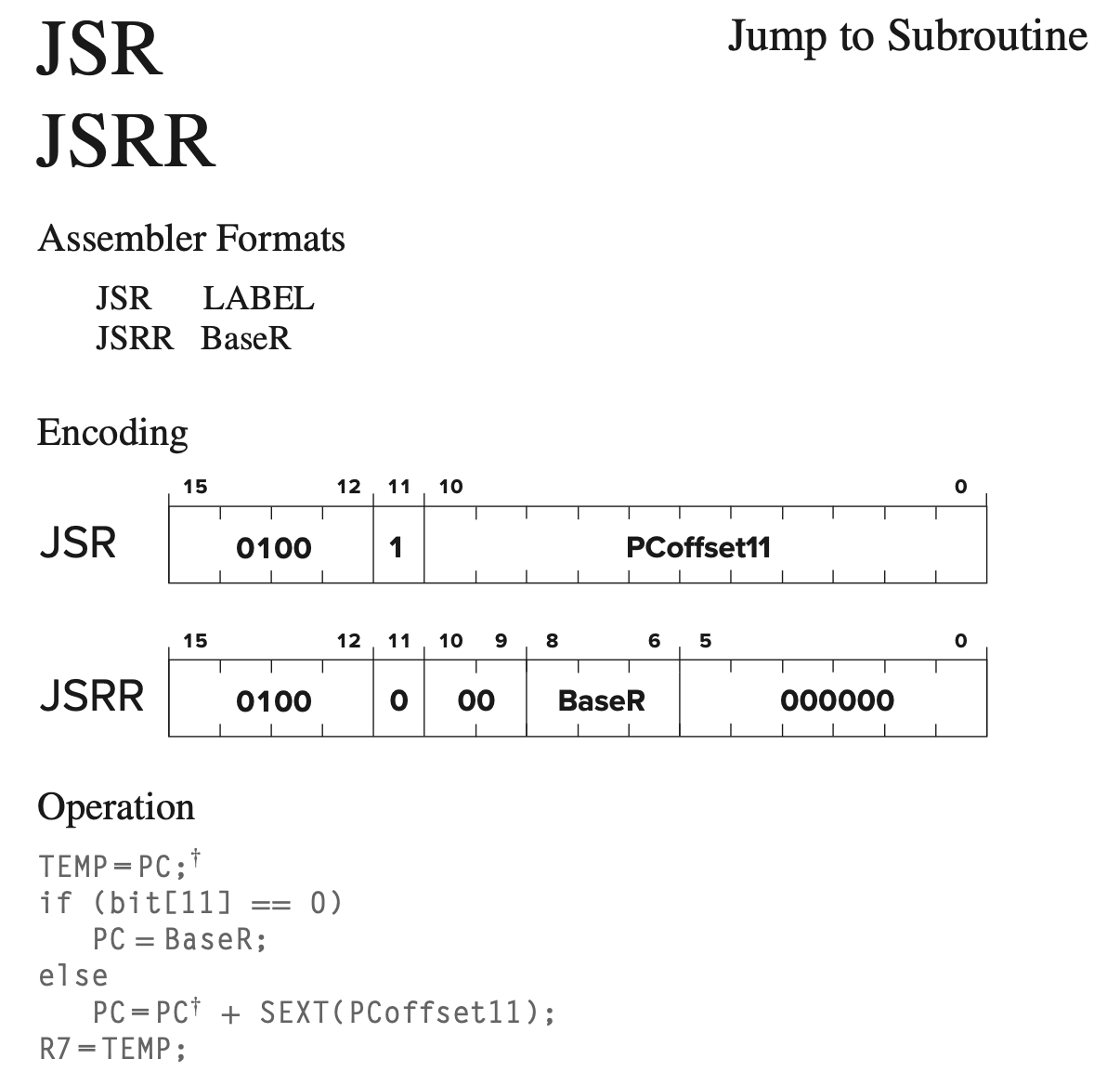
LC3 has five control instructions: **BR**, **JMP**, **JSR/JSRR**, TRAP and RTI. Both **BR** and **JMP** can be used to write loop statements. **JSR/JSRR** can be used to write subroutines.

**BR** is a Conditional Branch instruction that will change the value of PC based on the result of the conditional code test. The condition codes are specified by the state of bits [11:9]. There are eight combinations: BR, BRn, BRp, BRz, BRzp, BRnp, BRnz, BRnzp. If the condition is satisfied, the program branches to the location specified by adding the sign-extended PCoffset9 field to the incremented PC. Otherwise, PC remains unchanged, i.e., the program continues to execute the next instruction.



**JMP** instruction will **unconditionally jump** to the location specified by the contents of the base register which is identified by Bits[8:6]. **RET** is **JMP R7**.

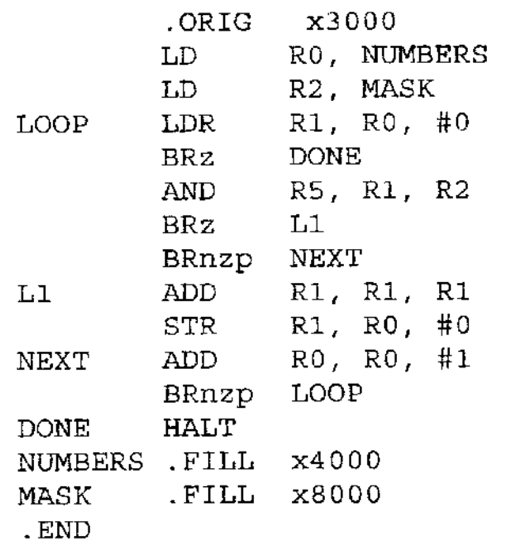
# JSR/JSRR will put the address of the instruction following JSR/JSRR into R7 and jump to subroutine. At the end of the subroutine, there should be a RET instruction.



# Lab Instruction

**Step 1: Download the LC-3 Simulator for Windows and unzip it to your local disk.**

**Step 2: Assume a sequence of integers is stored in consecutive memory locations, one integer per memory location, starting at the location x4000. The sequence terminates with the value x0000. What does the following program do?**

****

**Figure 1**

**Step 3: Type in the LC-3 assembly program in Figure 1, assemble it and run it in the simulator to verify its function.**

# Lab Exercise

1. Write a program to call a subroutine to print “Hello world”.

|  |
| --- |
| .ORIG x3000  MAIN LEA R2, MSG  JSR PRINT\_LOOP  STOP HALT  PRINT\_LOOP  LDR R1, R2, #0  BRz STOP  ADD R0, R1, #0  OUT  ADD R2, R2, #1  BRnp PRINT\_LOOP  MSG .STRINGZ "Hello world"  .END |

1. Write a program to check for ODD/EVEN number. Make it loops many times, halts when user enter ‘0’: (Assuming user only input single digit number 1~9, and exit when the input is ‘0’)

◦ Please enter a number: 2

◦ Thanks, this is an even number.

◦ Please enter a number: 9

◦ Thanks, this is an odd number.

◦ Please enter a number: 0

◦ Bye!

**Hints**: You can use PUTS to print a string to the screen and GETC to receive an input from the keyboard. You should define at least two subroutines, one for printing messages and the other for testing if a given number is odd or even.

|  |
| --- |
| .ORIG x3000  MAIN  LEA R0, PROMPT  PUTS  GETC  OUT  LD R1, ASCII\_ZERO  NOT R1, R1  ADD R1, R1, #1  ADD R2, R0, R1  BRz PRINT\_BYE  JSR TEST\_NUMBER    TEST\_NUMBER  AND R3, R2, #1  BRp PRINT\_ODD  BRz PRINT\_EVEN    PRINT\_ODD  LEA R0, ODD\_MSG  PUTS  JSR MAIN  PRINT\_EVEN  LEA R0, EVEN\_MSG  PUTS  JSR MAIN  PRINT\_BYE  LEA R0, BYE\_MSG  PUTS  HALT  PROMPT .STRINGZ "Please enter a number: "  ODD\_MSG .STRINGZ "\nThanks, this is an odd number.\n"  EVEN\_MSG .STRINGZ "\nThanks, this is an even number.\n"  BYE\_MSG .STRINGZ "\nBye!\n"  ASCII\_ZERO .FILL x0030  .END |

# Submission

Zip and Upload your source files and the .obj files (.asm and .obj) into ISpace. Please name the zip file with your student ID.