

## Notes: The LC-3 Computer (Control Instructions)

### A. Why?

Low-level programs use branch and jump instructions to make and implement decisions. TRAP instructions provide a way to access operating-system-level routines.

### B. Outcomes

By the end of the class you should

- Know how the LC-3 branch and jump instructions work.
- Know how the LC-3 TRAP instruction is used to read or write a character or halt the program.

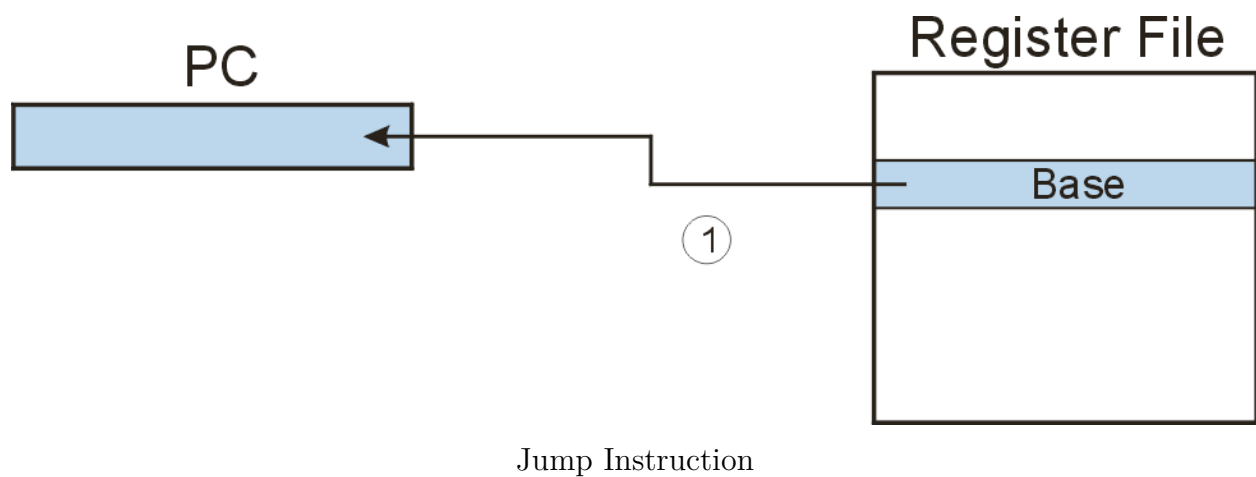
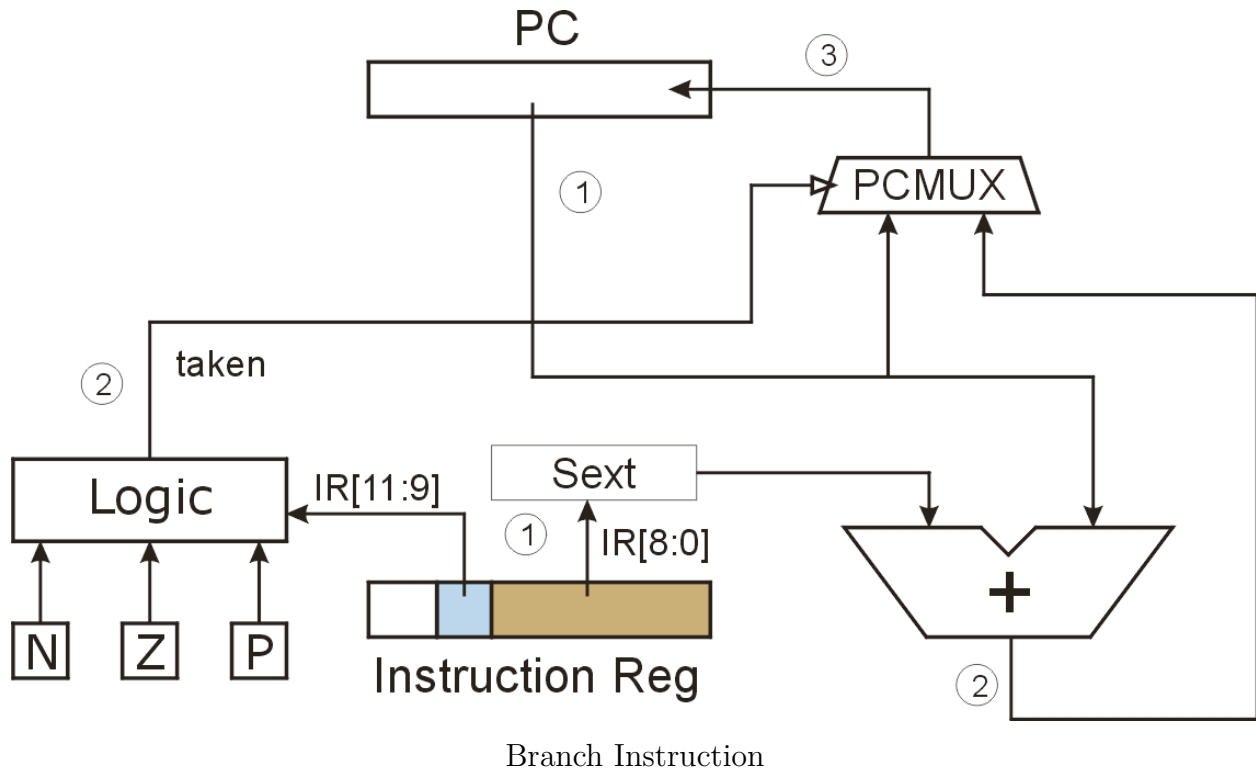
### C. Control Instructions

- Control instructions alter sequence of instructions being executed
  - Change the PC during the EXECUTE INSTRUCTION phase of instruction cycle.
  - Compare with PC change during FETCH INSTRUCTION phase of instr. cycle.
- Conditional (change made only under certain conditions) vs Unconditional (change always made). Also short-distance (“branch”) vs long-distance (“jump”) instructions.
- TRAP/service request — request action by operating system.
- Conditional branch (BR instruction)
  - If specified condition is true then  $PC \leftarrow PC + 9\text{-bit offset}$  [short jump]
  - Else do nothing (next instruction is the one after the branch instruction).
- LC-3 condition codes
  - Three 1-bit condition code registers (N, Z, P for  $<$ ,  $=$ ,  $>$  0).
  - Set by every instruction that writes a value into a register
- Branch instruction has 3-bit mask; jump if bitwise AND of NZP and mask  $\neq$  000.
  - Can have 0 mask bits = 1 (no-op instruction); 1 mask bit = 1 (test for  $<$ ,  $=$ , or  $>$  0); 2 mask bits = 1 (test for  $\leq$ ,  $\neq$ , or  $\geq$  0); 3 mask bits = 1 (unconditional branch)



- Jump instruction: Unconditional jump:  $PC \leftarrow$  Base register. [Can do long jump.]





**D. Examples of Branch Instruction**

- BR 0000 100 bbbbbbbbbb — Branch to PC+bbbbbbbbbb if N code is set
- BR 0000 010 bbbbbbbbbb — Branch to PC+bbbbbbbbbb if Z code is set
- BR 0000 001 bbbbbbbbbb — Branch to PC+bbbbbbbbbb if P code is set
- BR 0000 011 bbbbbbbbbb — Branch to PC+bbbbbbbbbb if Z or P code is set
- BR 0000 110 bbbbbbbbbb — Branch to PC+bbbbbbbbbb (regardless of cond. code).

## E. Sample Program With Loop

- Sum 12 integers starting at location x3100; pgm starts at location x3000.
- R1 = x3100;                   // R1 -> x3100...~~x310B~~ **x301C**  
   R3 = 0;                       // R3 = sum of M[x3100], M[x3101], ..., M[R1-1]  
   R2 = 12;                    // R2 = # Values yet to sum = x310C - R1  
   while R2 != 0 {  
       R4 = \*R1;            // R4 = M[R1] (R1 holds address of memory location)  
       R3 += R4;            // R3 = R3 + R4   Add current value to rolling sum  
       R1++;                // R1 = R1 + 1   Point to next value to add  
       R2--;                // R2 = R2 - 1   Decr. count of values to add  
   }

Addr	OpC	Instruction	Comments
x3000	LEA	1110 001 0111111111	R1 ← x3100 (PC+0xFF)
x3001	AND	0101 011 011 1 00000	R3 ← 0
x3002	AND	0101 010 010 1 00000	R2 ← 0
x3003	ADD	0001 010 010 1 01100	R2 ← 12
x3004	BR	0000 010 000000101	Loop: If Z, goto x300A (PC+5)
x3005	LDR	0110 100 001 000000	R4 = *R1
x3006	ADD	0001 011 011 0 00 100	R3 ← R3 + R4
x3007	ADD	0001 001 001 1 00001	++R1 (pointer)
x3008	ADD	0001 010 010 1 11111	--R2 (counter)
x3009	BR	0000 111 111111010	End loop: Goto x3004 (PC-6)