COMPUTER ORGANIZATION (IS F242)

LECT 18: LC 3 ARCHITECTURE

Control Instructions

- Used to alter the sequence of instructions by changing the PC
- LC-3 has 5 opcodes to break sequential flow
  - Conditional branch
  - Unconditional branch (jump)
  - Subroutine (function) call
  - TRAP
  - return from interrupt
- Conditional Branch
  - Branch is *taken* if a specified condition is true
    - signed offset is added to PC to yield new PC
  - Else, the branch is *not taken*
    - PC is not changed, points to the next sequential instruction

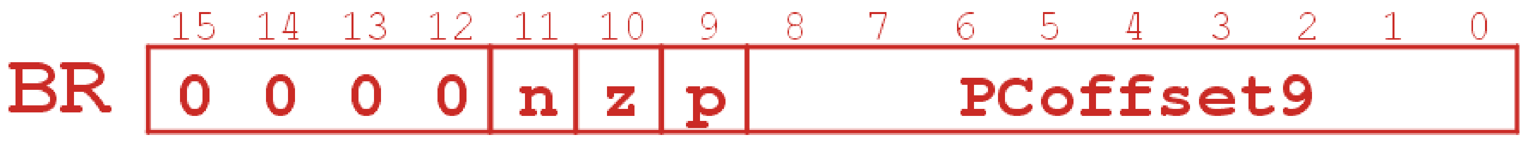
Control Instructions

- Unconditional Branch (or Jump)
  - Always changes the PC
- **TRAP**
  - Allows a programmer to get information into and out of the computer without fully understanding the intricacies of the input and output devices.
  - changes PC to the address of an OS "service routine"
  - routine will return control to the next instruction (after TRAP)

Condition Codes (Part of ISA)

- Allows the instruction sequencing to change on the basis of a previously generated result.
- LC-3 has three (single bit) condition code registers:
  - N – negative
  - Z – zero
  - P – positive (greater than zero)
- Set by any instruction that writes a value to a register
  - ADD, AND, NOT, LD, LDR, LDI, LEA
- Exactly one will be set at all times
  - Based on the last instruction that altered a register

BR (PC-Relative)



n -- negative
z -- zero
p -- positive

Bits [11], [10] & [9] correspond to the three condition codes.

The condition codes are used by the conditional branch instruction to determine whether to change the instruction flow.

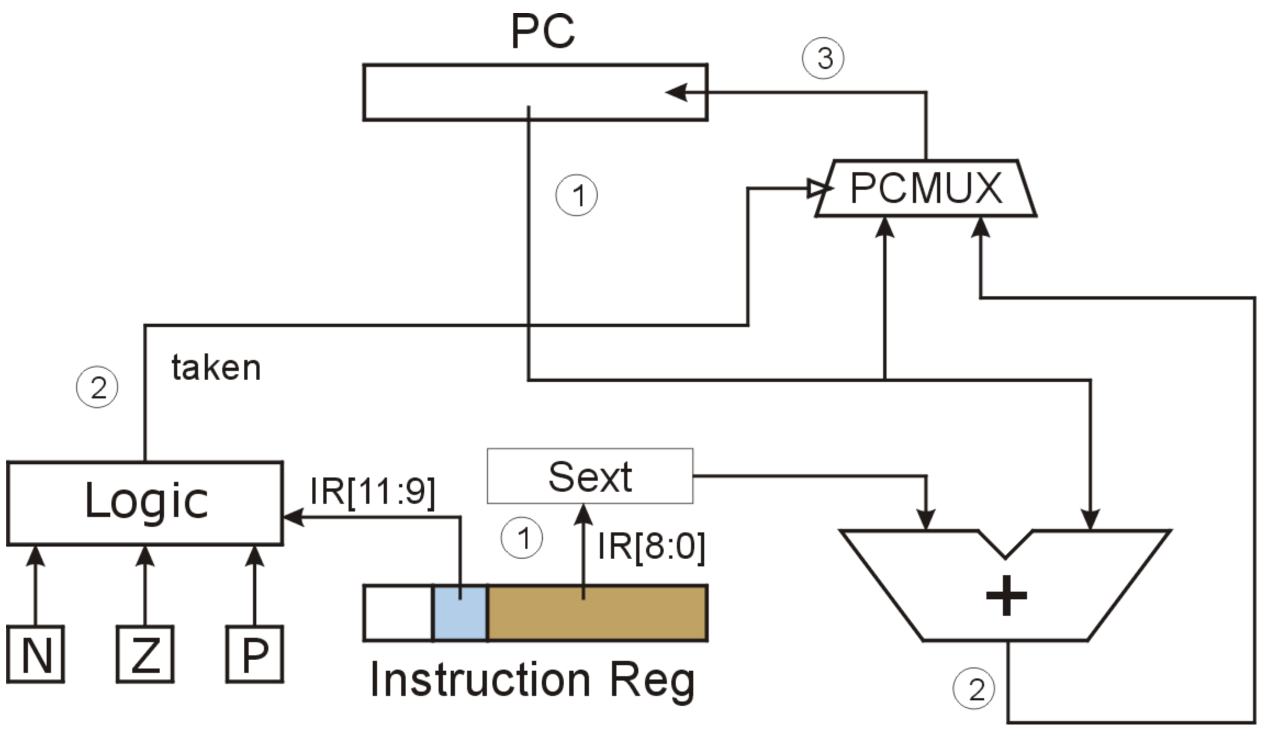
Instruction Cycle

- Fetch & Decode is same as any other instruction.
- Evaluate Address is same as LD & ST.
  - Address is computed by adding the incremented PC to the 16 bits (SEXT (IR[8:0])).
- During Execute phase, condition codes are examined.
  - If bit [11] is 1 condition code N is examined.
  - If bit [10] is 1 condition code Z is examined.
  - If bit [9] is 1 condition code P is examined.
  - If any of bits [11:9] are 0, corresponding condition codes are not examined.

Instruction Cycle

- If any of the condition codes examined are in state 1 then
  - PC is loaded with the address obtained in the Evaluate Address phase.
- If none of the condition codes examined are in state 1, the PC is left unchanged.
  - In that case, in the next instruction cycle, the next sequential instruction will be fetched.

BR (PC-Relative): Data Path



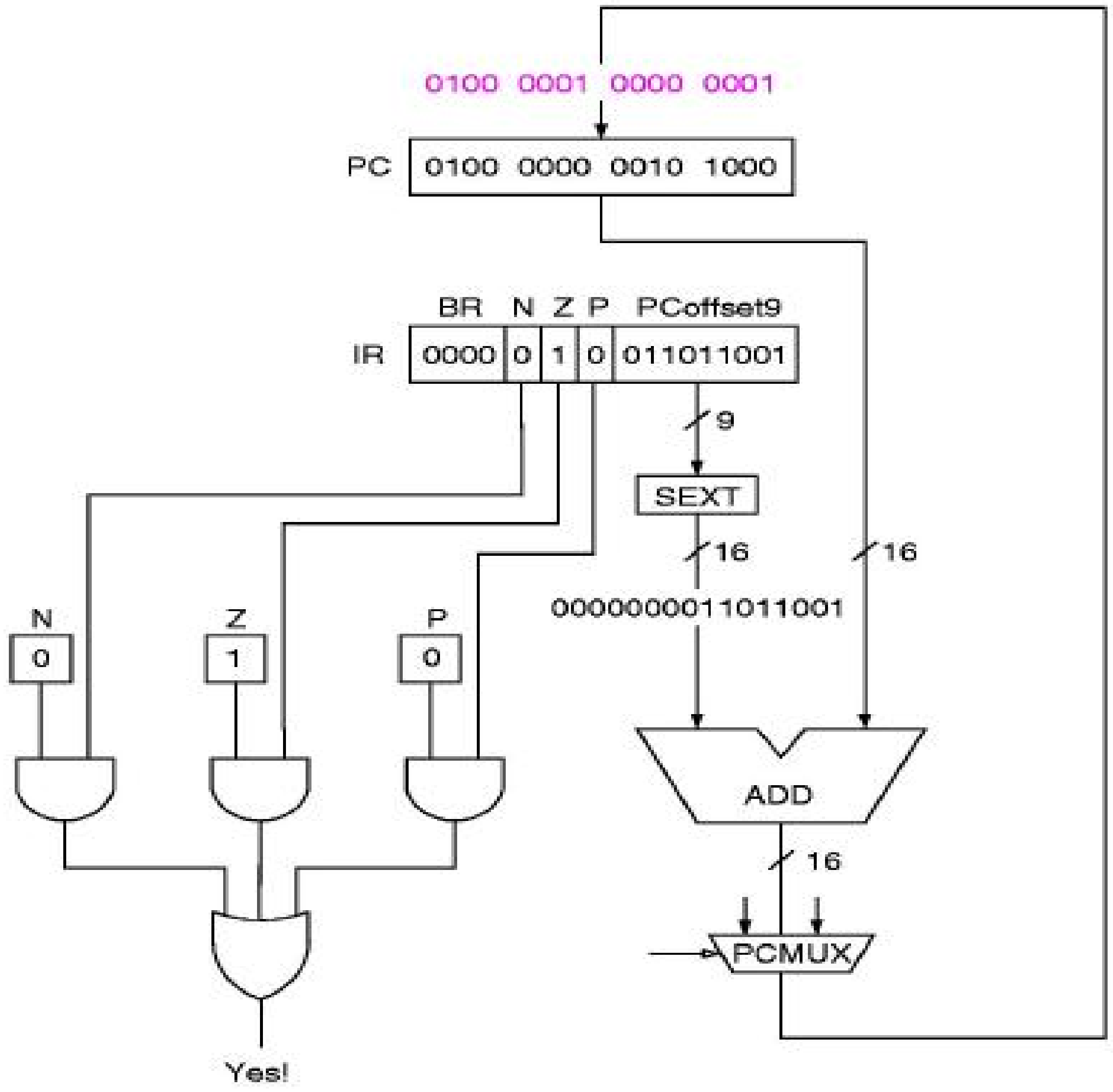
If the condition code used in BR is set to 1 the new PC contents will be the addition of signed offset (IR[8:0]) to current PC value (Branch is Taken)

Else PC contains the next sequential address (Branch Not Taken)

What happens if bits [11:9] are all zero? All one?

BR Data Path

BRz x0D9



Unconditional Branching

- If all three bits [11:9] are 1, then All three condition codes are examined.
- Since the last result stored into a register had to be either positive, negative or zero.
- One of the three condition codes must be in state 1.
- So PC is loaded with the address obtained in the Evaluate address phase.
- This is called Unconditional branch (jump) as the instruction flow is changed unconditionally

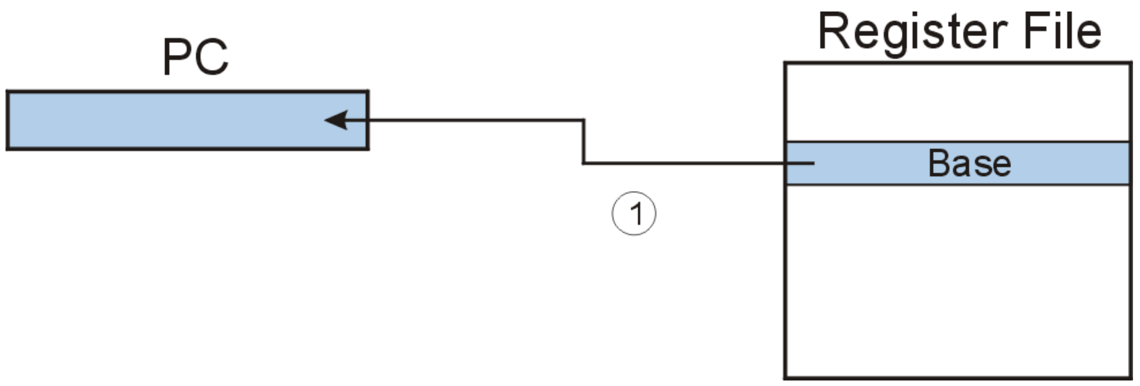
Unconditional Branching

- Limitation of conditional Branch
  - Next instruction executed must be with in the range +256 to -255 locations from the current branch instruction.
- What if the program wants to next instruction that is 1000 locations from the current instruction?
- LC – 3 provides JMP (Jump) instruction to do this job

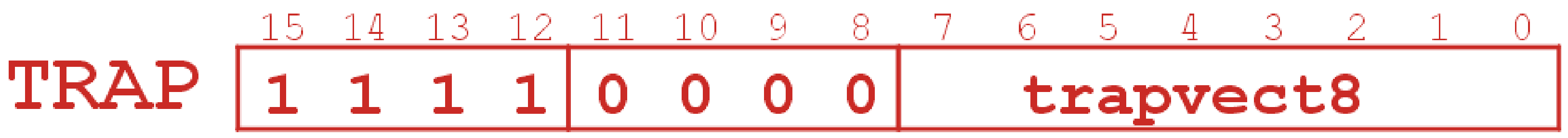
JMP (Register)

- Jump is an unconditional branch -- ***Always*** taken.
  - Target address is the contents of a register.
  - Allows any target address.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JMP	1	1	0	0	0	0	0	Base			0	0	0	0	0	0



TRAP

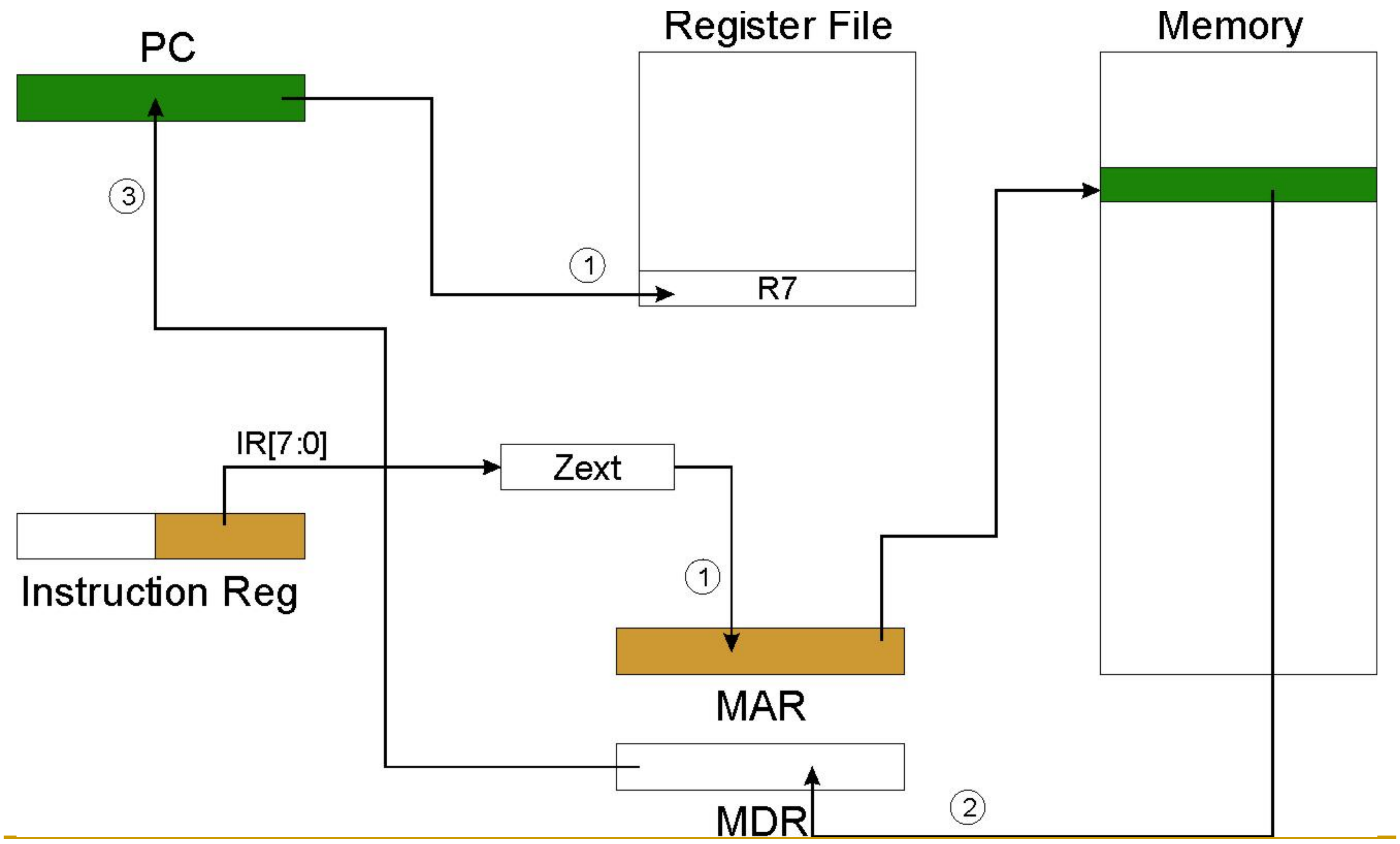


- Changes PC to a memory address that is part of the OS so that OS will perform some task on behalf of the program that is executing.
- It calls a **service routine**, identified by 8-bit "trap vector."

vector	routine
x23	**input a character from the keyboard**
x21	**output a character to the monitor**
x25	**halt the program**

- When routine is done, PC is set to the address of the instruction following the TRAP instruction & the program continues.

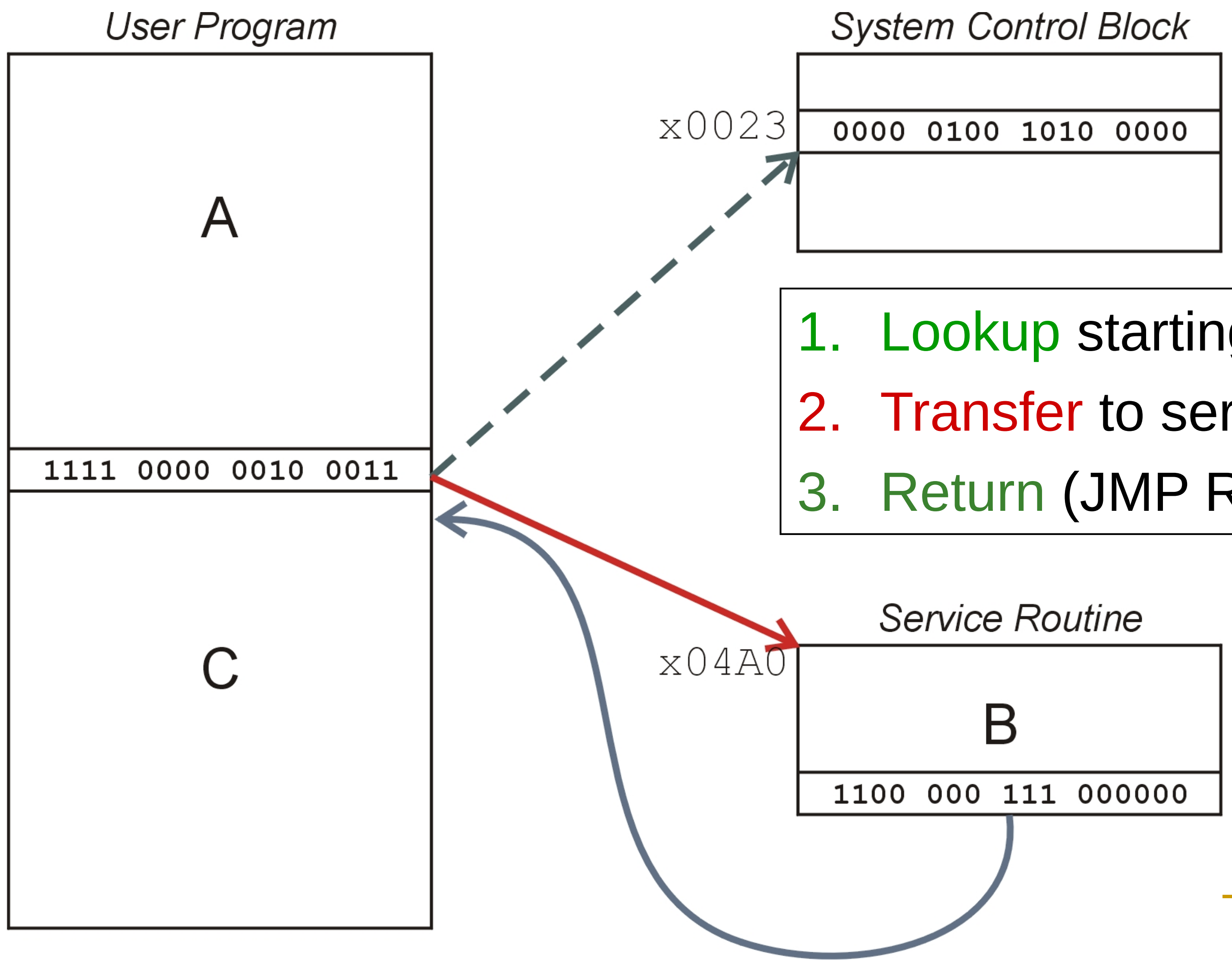
TRAP


PC
Register File
Memory
3
1
R7
IR[7:0]
Zext
Instruction Reg
1
MAR
MDR
2

RET (JMP R7)

- How do we transfer control back to instruction following the TRAP?
- We saved old PC in R7.
  - JMP R7 gets us back to the user program at the right spot.
  - LC-3 assembly language lets us use RET (return) in place of “JMP R7”.
- Must make sure that service routine does not change R7, or we won’t know where to return.

TRAP Mechanism Operation



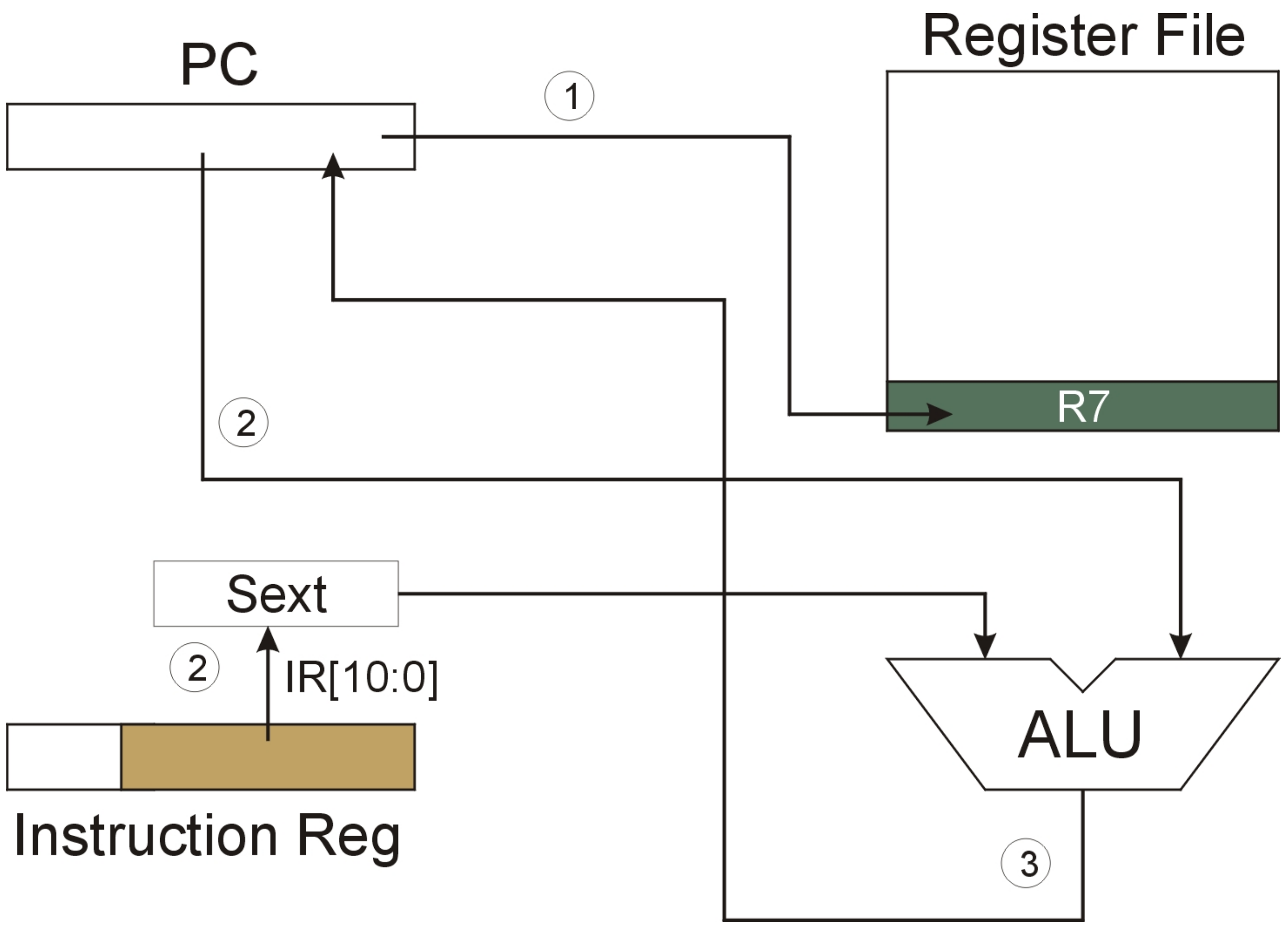
1. Lookup starting address.
2. Transfer to service routine.
3. Return (JMP R7).

JSR Instruction

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSR	0	1	0	0	1	PCoffset11										

- Jumps to a location (like a branch but unconditional),
  and saves current PC (addr of next instruction) in R7.
  - saving the return address is called “linking”
  - target address is PC-relative (PC + Sext(IR[10:0]))
  - bit 11 specifies addressing mode
    - if =1, PC-relative: target address = PC + Sext(IR[10:0])
    - if =0, register: target address = contents of register IR[8:6]

JSR



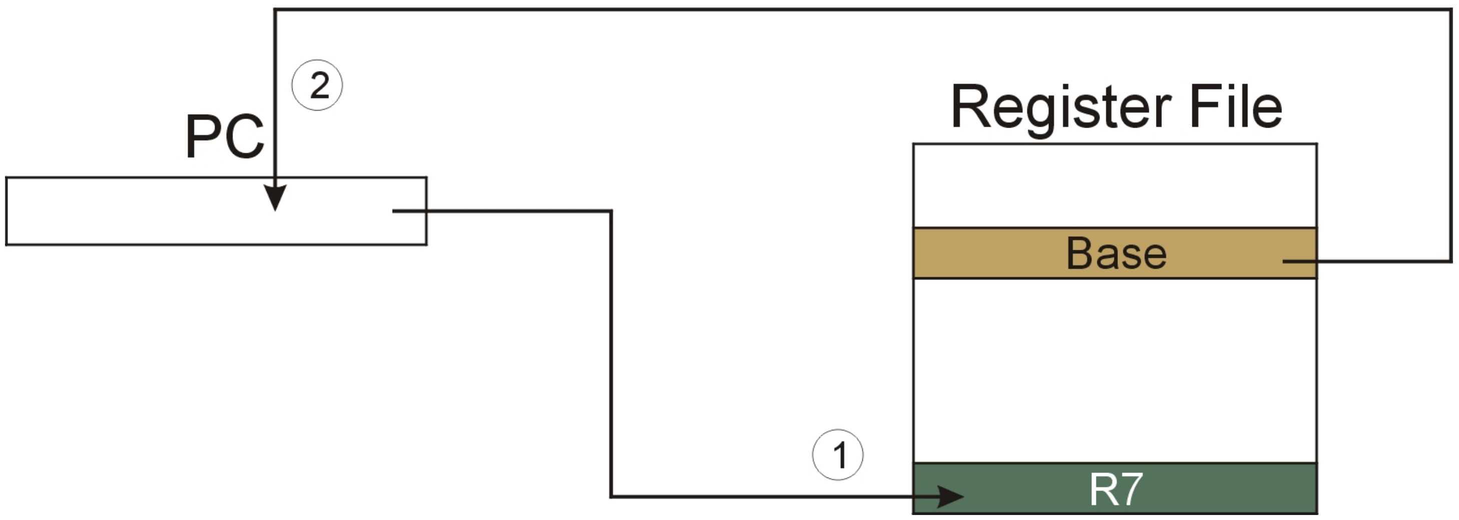
JSRR Instruction

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSRR	0	1	0	0	0	0	0	Base			0	0	0	0	0	0

- Just like JSR, except Register addressing mode.
  - target address is Base Register
  - bit 11 specifies addressing mode

- What important feature does JSRR provide that JSR does not?

JSRR



PC
2
Register File
Base
1
R7