COMPUTER ORGANIZATION (IS F242)

LECT 15: LC 3 ARCHITECTURE

Instruction Set Architecture of LC-3

- ISA = All of the *programmer-visible* components and operations of the computer
- ISA - Interface between what the software commands and what the hardware actually carried out
- ISA Specifies
  - Memory Organization
  - Register set
  - Instruction set
    - Opcodes
    - Data types
    - Addressing modes

Instruction Set Architecture of LC-3

- Memory organization
  - Address space -- How may locations can be addressed?
    - Address space: 2^{16} locations (16-bit addresses)
  - Addressability -- How many bits per location?
    - Addressability: 16 bits (referred to as 1 word)
    - LC – 3 is Word addressable
- Register set
  - How many? What size?
  - Eight general-purpose registers (GPR) : R0 - R7 Each 16 bits wide
    - Temporary storage, accessed in less time
    - Accessing memory generally takes longer time
    - Registers are uniquely identifiable with 3 bit register number
  - Other registers
    - Not directly addressable, but used by (and affected by) instructions
    - PC (program counter), condition codes

Instruction Set Architecture of LC-3 Cont…

- Instruction set
  - Made up of opcode and operands
  - ISA is defined by its set of opcodes, data types and addressing modes
  - Opcodes
    - 15 opcodes (Each is identified by unique 4-bit code)
    - Specified by bits [15:12] of the instruction
    - ***Operate*** instructions:
      - Processes information
      - ADD, AND, NOT
    - ***Data movement*** instructions:
      - Move information between memory and the registers and between registers/memory and input/output devices
      - LD, LDI, LDR, LEA, ST, STR, STI
    - ***Control*** instructions:
      - Change the sequence of instructions that will be executed.
      - BR, JSR/JSRR, JMP, RTI, TRAP

Instruction Set Architecture of LC-3 Cont…

- Data types
  - Representation on which opcodes operate
  - 16-bit 2's complement integer
- Addressing modes
  - Determines where the operands are located
    - (In Memory, In a register or as a part of the instruction)
  - How is the location of an operand specified?
    - Memory addresses: *PC-relative*, *indirect*, *base+offset*
    - Register addresses
    - Immediate addresses (As a part of instruction) also called literal operands
- Condition codes
  - 3 single bit registers called N, Z and P corresponding to Negative, Zero and Positive
  - Each time a GPR is written, the N, Z and P registers are individually set to 0 or 1 based on GPR value.
- *ISA provides all information needed to write a program in machine language*

Addressing Modes

- LC-3 supports five addressing modes
  - Immediate (or literal)
  - Register
  - PC-relative
  - Indirect
  - Base + Offset
- Last three addressing modes are memory addressing modes

Operate Instructions

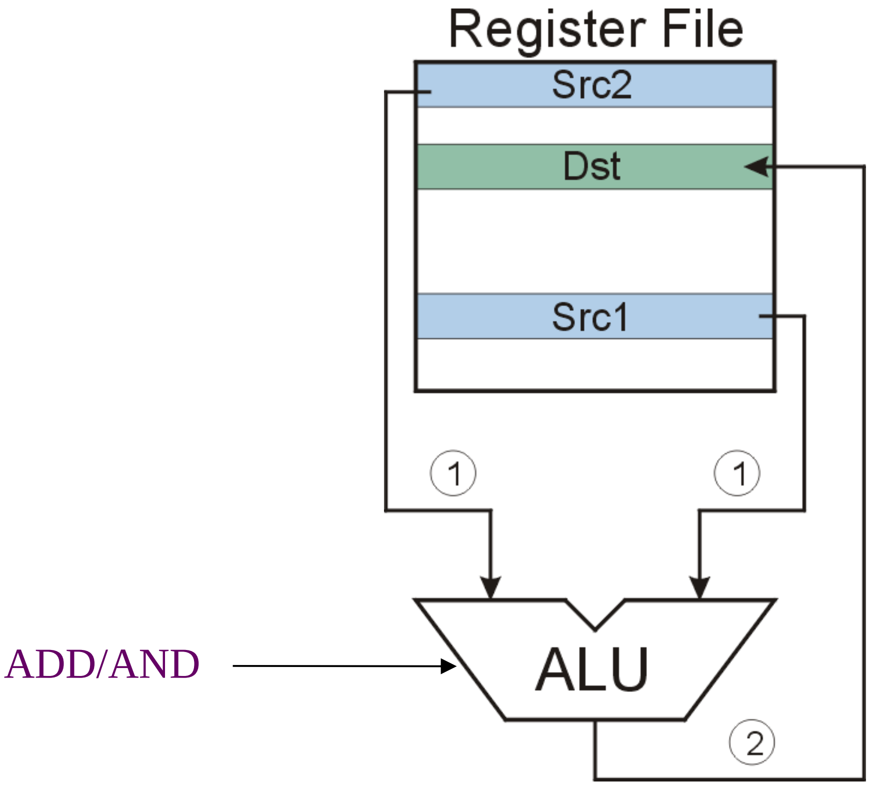
- Only three operate Instructions:
  - ADD (opcode: 0001)
  - AND (opcode: 0101)
  - NOT (opcode: 1001) - Unary operation
- Immediate and Register addressing modes
- Source and destination operands are registers
  - These instructions *do not* reference memory.
  - ADD and AND can use "*immediate*" mode,
    where one operand is hard-wired into the instruction.
- Show dataflow diagram with each instruction.
  - Illustrates *when* and *where* data moves to accomplish the desired operation

ADD/AND (Register)

this zero means “register mode”

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD	0	0	0	1	Dst			Src1			0	0	0	Src2		

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AND	0	1	0	1	Dst			Src1			0	0	0	Src2		



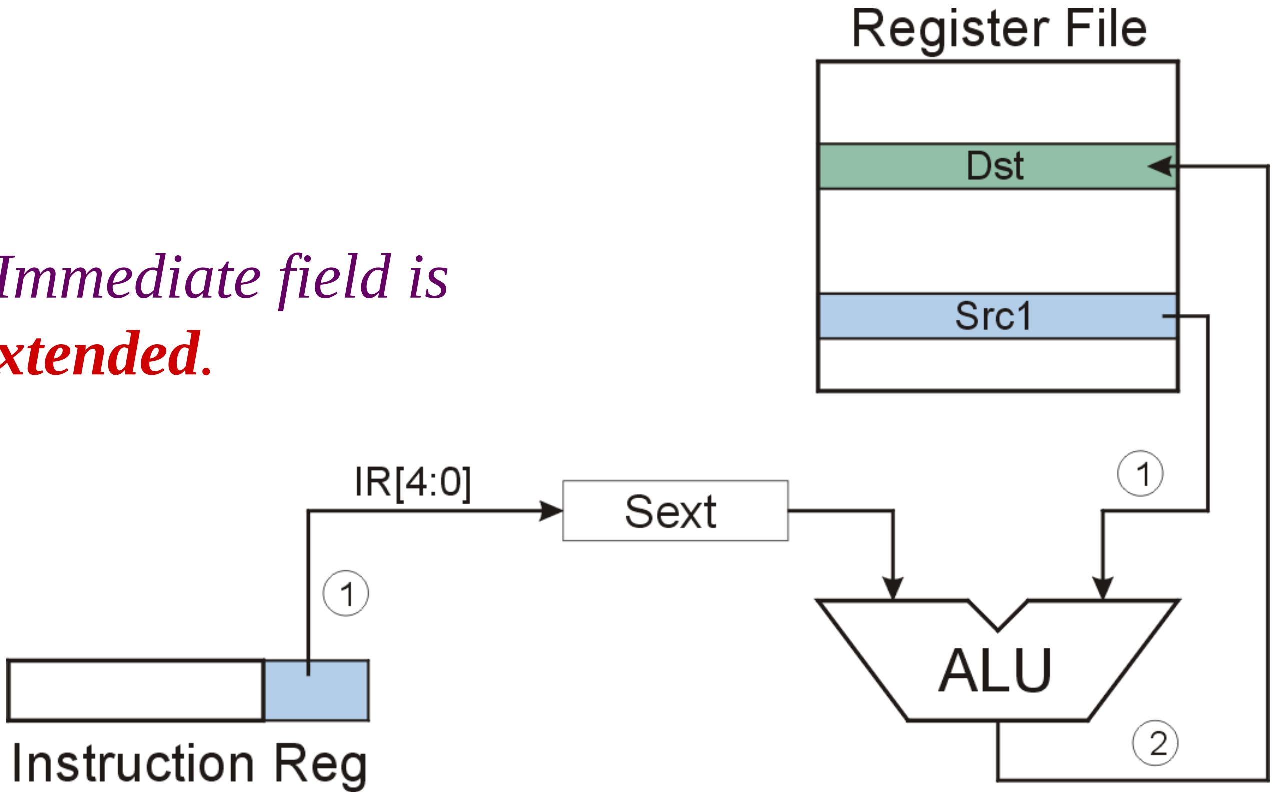
ADD/AND (Immediate)

this one means “immediate mode”

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD	0	0	0	1	Dst			Src1			1	Imm5				

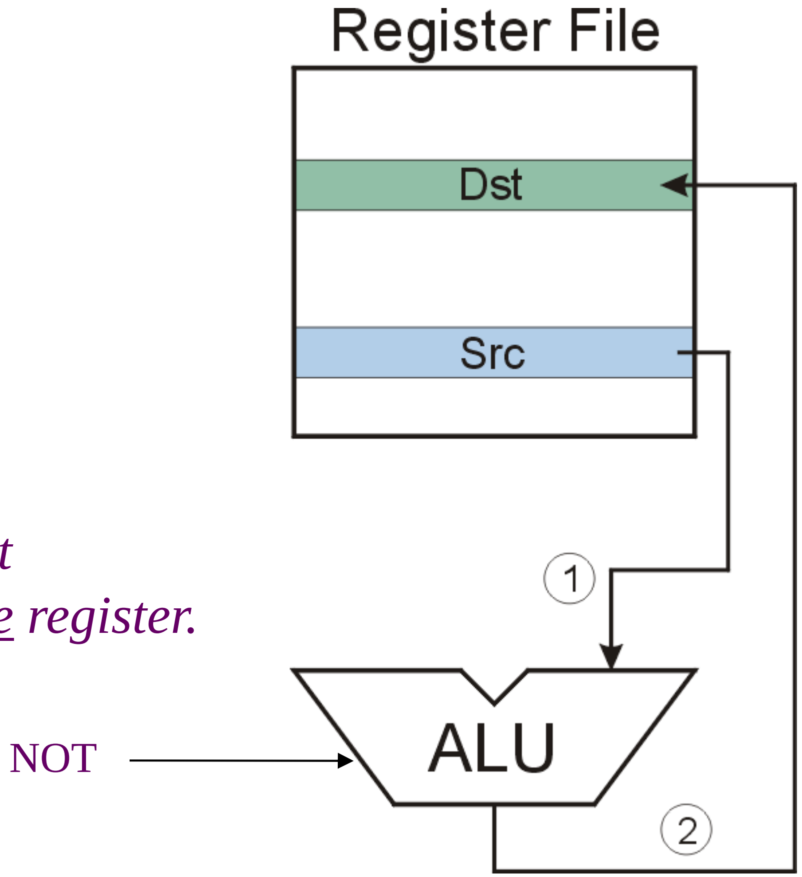
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AND	0	1	0	1	Dst			Src1			1	Imm5				

Note: Immediate field is ***sign-extended*****.**



NOT (Register)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT	1	0	0	1	Dst			Src			1	1	1	1	1	1



Note: Src and Dst could be the <u>same</u> register.

Operating Instructions

- With only ADD, AND, NOT…

1. How do we subtract?

   i.e. A - B

2. How do we initialize a register to zero?

3. How do we copy from one register to another?

4. How do we OR?

A - B

Assuming the values A and B are in R0 and R1

1 0 0 1	0 0 1	0 0 1	1 1 1 1 1 1
NOT	R1	R1	

R1← NOT(B)

0 0 0 1	0 1 0	0 0 1	1	0 0 0 0 1
ADD	R2	R1		1

R2← - B

0 0 0 1	0 1 0	0 0 0	0	0 0	0 1 0
ADD	R2	R0			R2

R2← A + (-B)

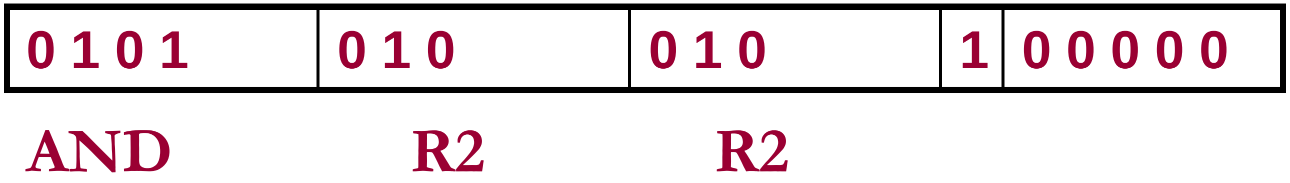
Initialize a register with Zero

0101	010	010	1	00000
AND	R2	R2		

R2← 0

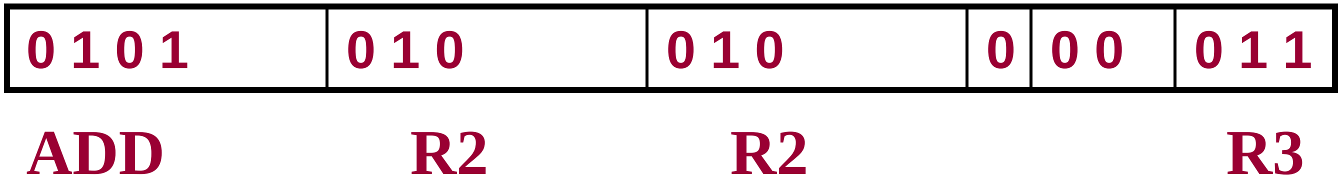
Register R2 is initialized to zero

Copy from one register to Another



0101	010	010	1	00000
AND	R2	R2		

R2←0



0101	010	010	0	00	011
ADD	R2	R2			R3

R2←R2+R3

Contents of R3 are copied to R2

Increment a Register by 1

0 0 0 1	1 1 0	1 1 0	1	0 0 0 0 0 1
ADD	R6	R6		Imm

R6← R6 + 1

Register R6 is incremented by 1