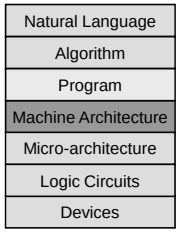
These notes are based on Dr Delmas' and many other people's notes

Chapter 5

The LC-3



Agenda & Reading

- The LC3 ISA
- LC3 operate instructions
- LC3 data movement instructions
- LC3 control instructions

- Read Chapter 5 of the textbook

Instruction Set Architecture

- ISA = All of the *programmer-visible* components and operations of the computer
  - memory organization
    - address space -- how may locations can be addressed?
    - addressibility -- how many bits per location?
  - register set
    - how many? what size? how are they used?
  - instruction set
    - opcodes
    - data types
    - addressing modes
- ISA provides all information needed for someone that wants to write a program in machine language (or translate from a high-level language to machine language).

LC-3 Overview: Memory and Registers

Memory
- address space: 2^{16} locations (16-bit addresses)
- addressability: 16 bits

Registers
- temporary storage, accessed in a single machine cycle
  - accessing memory generally takes longer than a single cycle
- eight general-purpose registers: R0 - R7
  - each 16 bits wide
  - how many bits to uniquely identify a register?
- other registers
  - not directly addressable, but used by (and affected by) instructions
  - PC (program counter), condition codes

LC-3 Overview: Instruction Set

Opcodes
- 15 opcodes
- ***Operate*** instructions: ADD, AND, NOT
- ***Data movement*** instructions: LD, LDI, LDR, LEA, ST, STR, STI
- ***Control*** instructions: BR, JSR/JSRR, JMP, RET, TRAP
- some opcodes set/clear *condition codes*, based on result:
  - N = negative, Z = zero, P = positive (> 0)

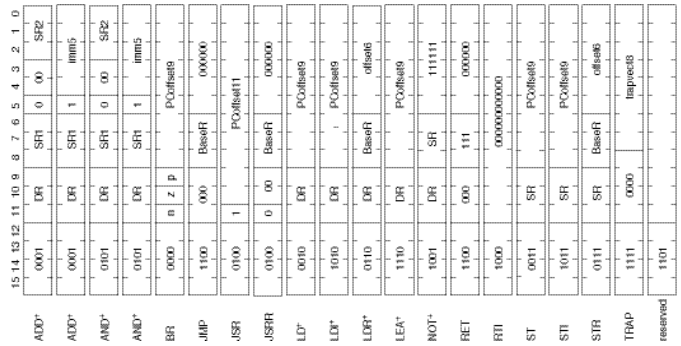
Data Types
- 16-bit 2's complement integer

Addressing Modes
- How is the location of an operand specified?
- non-memory addresses: *immediate*, *register*
- memory addresses: *PC-relative*, *indirect*, *base+offset*

LC 3 Instructions

LC-3 Instruction word: 16 bits
- Opcode: specifies what the instruction does
  - IR[15:12]: 4 bits allow 16 instructions
- Operands: what the instruction acts on
  - IR[11:0]: contains specifications for:
  - Registers: 8 GPRs (i.e. each requires 3 bits for addressing)
  - Address Generation bits: Offset (11 or 9 or 6 bits) (more later)
  - Immediate value: 5 bits

Full-LC3 Instruction set



Agenda

- The LC3 ISA
- **LC3 operate instructions**
  - **AND, ADD, NOT**
  - **Addressing mode: register, immediate**
- LC3 data movement instructions
- LC3 control instructions

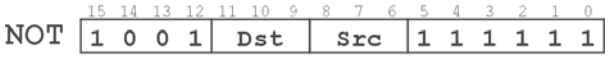
Operate Instructions

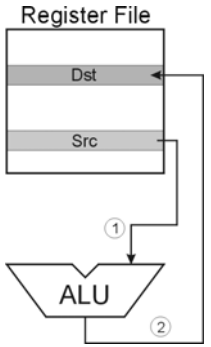
Only three operations: ADD, AND, NOT

Source and destination operands of these instructions are registers
- These instructions *do not* reference memory.
- ADD and AND can use "immediate" mode, where one operand is stored as part of the instruction.

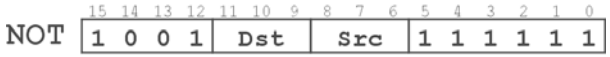
Will show dataflow diagram with each instruction.
- illustrates *when* and *where* data moves to accomplish the desired operation

NOT





Note: Src and Dst could be the same register.



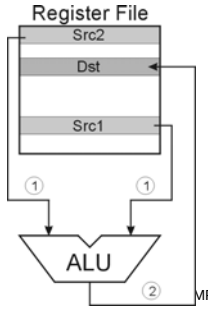
- 1001 001 010 111111

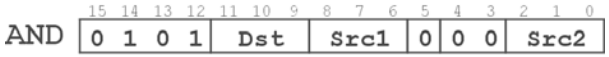
- Opcode: 1001 (NOT)
- Dst: 001 (R1)
- Src: 010 (R2)
- [5..0] are set to 1

- Assume the contents of R2 is 1
- After the instruction is executed, the contents in R1 are xFFFE
- The value stored in R2 is unchanged, i.e. 1

ADD/AND (register)

this zero means "register mode"





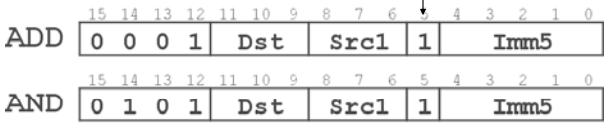
- 0101 100 010 0 00 011

- Opcode: 0101 (AND)
- Bit 5 is 0 to indicate that all the operands are in registers
- Dst: 100 (R4)
- Src1: 010 (R2)
- Src2: 011 (R3)

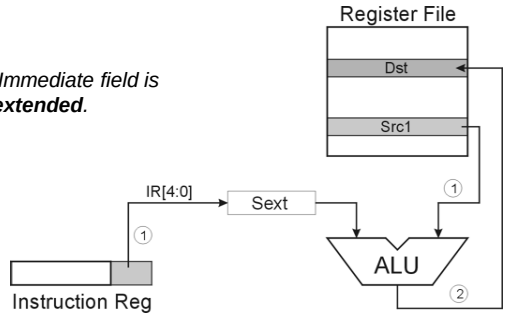
- Assume the contents of R2 is 1 and the contents of R3 is 3
- After the instruction is executed, the value in R4 is 1
- The values in R2 and R3 are unchanged

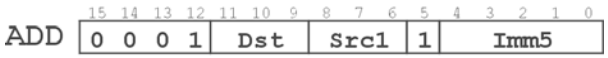
ADD/AND (Immediate)

this one means "immediate mode"



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





- 0001 110 101 1 00011

- Opcode: 0001 (ADD)
- Bit 5 is 1 to indicate [4..0] is a value operand
- Dst: 110 (R6)
- Src1: 101 (R5)
- Imm5: 00011 (i.e. 3)

- Assume the contents of R5 is 1
- After the instruction is executed, the value in R6 is 4
- The value in R5 is unchanged

Using Operate Instructions

With only ADD, AND, NOT…

– How do we subtract?

– How do we copy from one register to another?

– How do we initialize a register to zero?

– How do we left-shift a number?

Agenda

- The LC3 ISA
- LC3 operate instructions
- **LC3 data movement instructions**
  - **load, store, etc.**
  - **addressing mode: PC-relative mode, base+offset mode, indirect mode**
- LC3 control instructions

Data Movement Instructions

Load -- read data from memory to register
- LD: PC-relative mode
- LDR: base+offset mode
- LDI: indirect mode

Store -- write data from register to memory
- ST: PC-relative mode
- STR: base+offset mode
- STI: indirect mode

Load effective address -- compute address, save in register
- LEA: immediate mode
- *does not access memory*

PC-Relative Addressing Mode

Want to specify address directly in the instruction
- But an address is 16 bits, and so is an instruction!
- After subtracting 4 bits for opcode and 3 bits for register, we have 9 bits available for address.

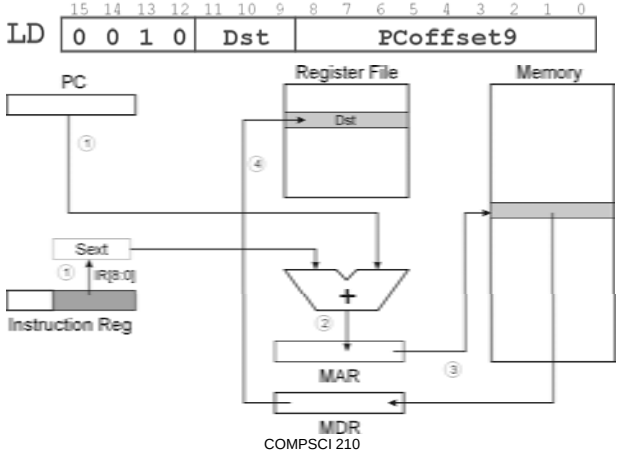
Solution:
- Use the 9 bits as a *signed offset* from the current PC.

9 bits: $-256 \le \text{offset} \le +255$

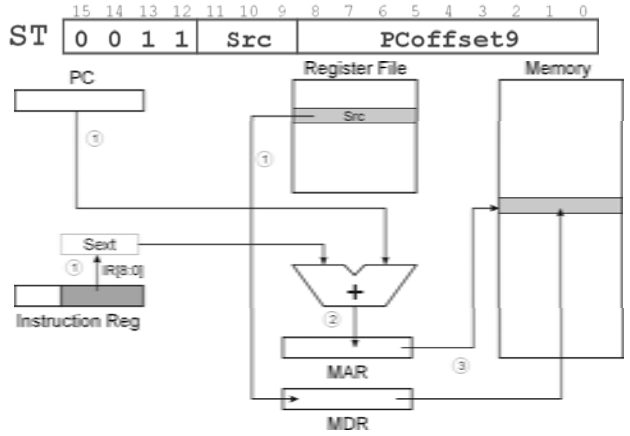
Can form any address X, such that: $PC - 256 \le X \le PC + 255$

Remember that PC is incremented as part of the FETCH phase;
This is done before the EVALUATE ADDRESS stage.
PC is the address of the NEXT instruction

LD (PC-Relative)



ST (PC-Relative)



- 0010 111 000011000
- Opcode: 0010 (LD)
- Dst: 111 (R7)
- PCOffset9: 000011000 (0x18)
- Assume the value of PC is 0x3001.
  - The address of the location that stores the value to be loaded is PC+PCOffset9 = 0x3019
- Assume that the content of location 0x3019 is 0x123
- After the instruction is executed, the value in R7 is 0x123

Indirect Addressing Mode

With PC-relative mode, we can only address data within 256 words of the instruction.

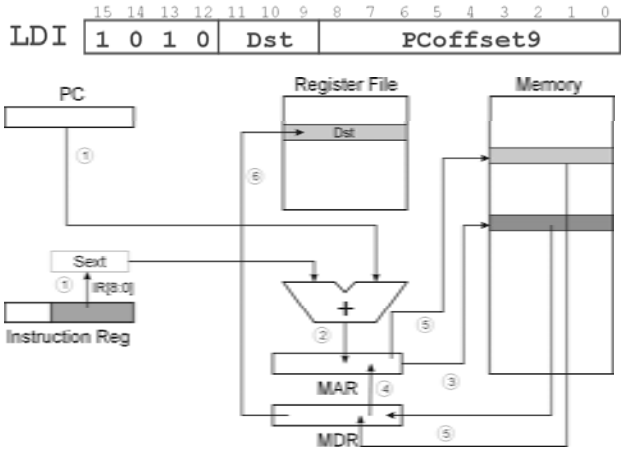
- What about the rest of memory?

Solution #1:

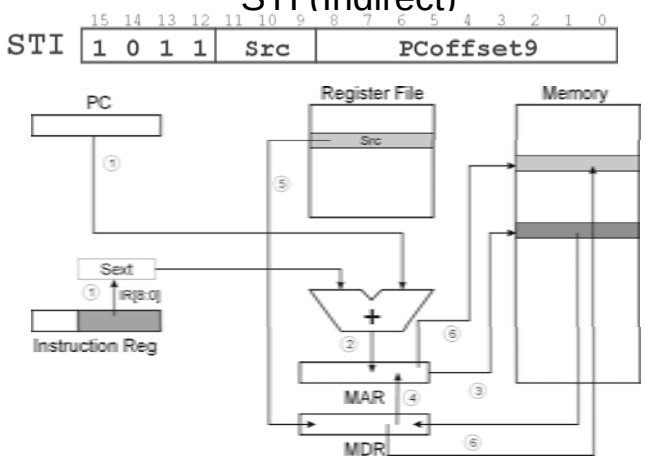
- Read address from memory location, then load/store to that address.

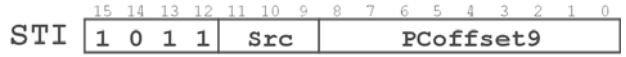
First address is generated from PC and IR (just like PC-relative addressing), then content of that address is used as target for load/store.

LDI (Indirect)



STI (Indirect)





- 1011 001 000011010

- Opcode: 1011 (STI)
- Src: 001 (R1)
- PCOffset9: 000011010 (0x1A)

- Assume the value of PC is 0x3003.
  - The address of the location that holds the address where the value will be stored is PC+PCOffset9 = 0x301D
- Assume the value stored at location 0x301D is 0x3000
- Assume that the content of R1 is 0x123
- After the instruction is executed, the value stored at 0x3000 is 0x123

Base + Offset Addressing Mode

With PC-relative mode, we can only address data within 256 words of the instruction.

- What about the rest of memory?

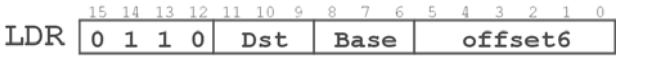
Solution #2:

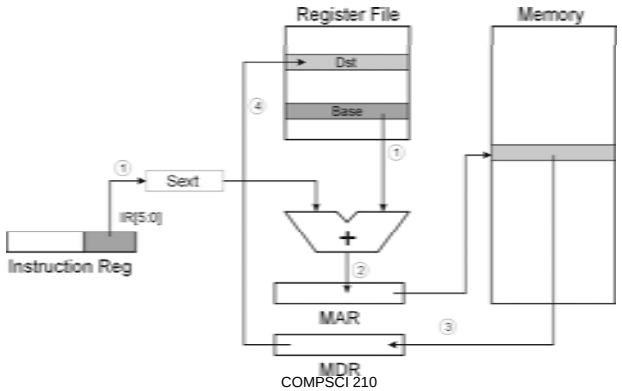
- Use a register to generate a full 16-bit address.

4 bits for opcode, 3 for src/dest register,
3 bits for *base* register
remaining 6 bits are used as a *signed offset*.

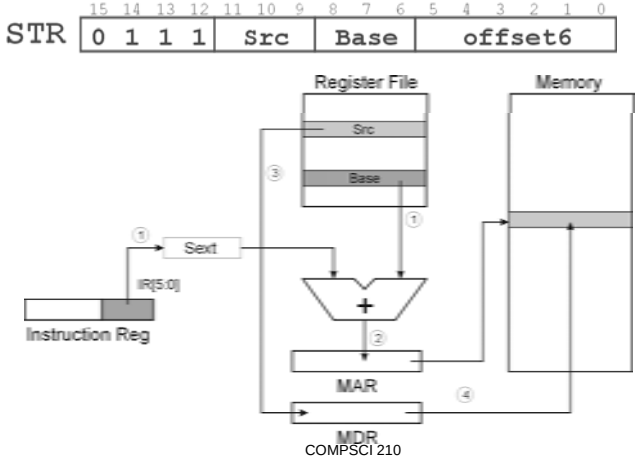
- **Offset is *sign-extended* before adding to base register.**

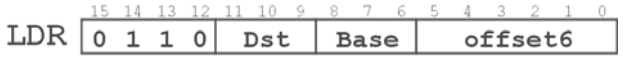
LDR (Base+Offset)





STR (Base+Offset)





- 0110 010 001 000010

- Opcode: 0110 (LDR)
- Dst: 010 (R2)
- Base: 001 (R1)
- Offset6: 000010 (0x2)

- Assume the value in R1 is 0x301E
- The address of the location that stores the value to be loaded is BaseReg+Offset6 = 0x3020
- Assume the value stored at location 0x3020 is 0x2345
- After the execution of the instruction, the value in R2 is 0x2345

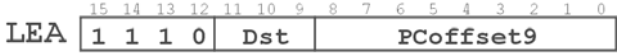
Load Effective Address

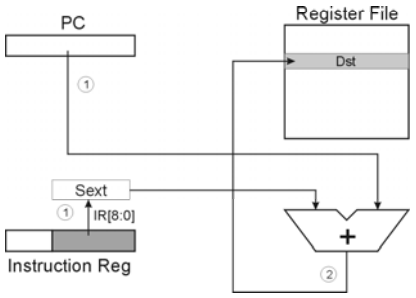
Want to load the address of a memory location into a register

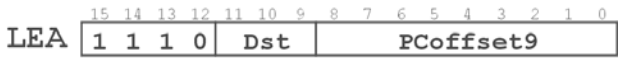
Computes address like PC-relative (PC plus signed offset) and stores the result into a register.

Note: The ***address*** is stored in the register, not the contents of the memory location.

LEA (Immediate)







- 1110 001 000011101

- Opcode: 1110 (LEA)
- Dst: 001 (R1)
- PCOffset9: 000011101 (0x1D)

- Assume the value of PC is 0x3001
- The address to be loaded is PC + PCOffset9 = 0x301E
- After the execution of the instruction, the value stored in R1 is 0x301E

Agenda

- The LC3 ISA
- LC3 operate instructions
- LC3 data movement instructions
- **LC3 control instructions**
  - **conditional, un-conditional branch**
  - **condition code**

Control Instructions

Used to alter the sequence of instructions
(by changing the Program Counter)

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

Unconditional Branch (or Jump)
- always changes the PC

TRAP
- changes PC to the address of an OS “service routine”
- routine will return control to the next instruction (after TRAP)

Condition Codes

LC-3 has three 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 <u>one</u> will be set at all times
- Based on the last instruction that altered a register

Branch Instruction

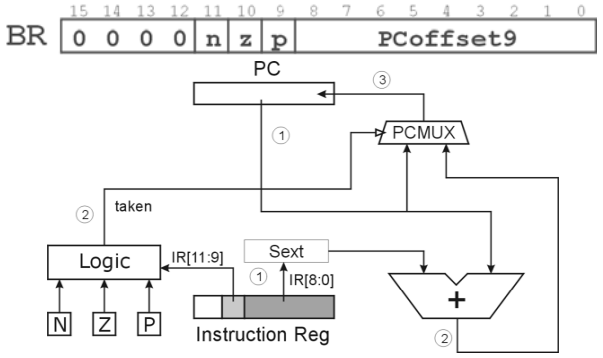
Branch specifies one or more condition codes.

If the specified bit is set, the branch is taken.

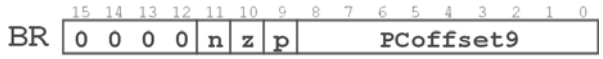
- PC-relative addressing:
  target address is made by adding signed offset (IR[8:0])
  to current PC.
- Note: PC has already been incremented by FETCH stage.
- Note: Target must be within 256 words of BR instruction.

If the branch is not taken,
the next sequential instruction is executed.

BR (PC-Relative)



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



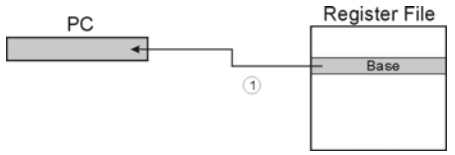
- To check whether the value stored in a register by the previous instruction is 0, the following instruction can be used
  0000 010 000011111
- Opcode: 0000 (BR)
- nzp: 010 (check z)
- PCOffset9: 000011111 (0x1F)

- Assume the value of PC is x3001
- If Z is 0, the instruction at 0x3001 will be executed
- If Z is 1, PC = PC + PCOffset9 = x3020
  - The instruction at address 0x3020 will be executed

JMP (Register)

Jump is an unconditional branch -- *always* taken.

- Target address is the contents of a register.
- Allows any target address.



- 1100 000 010 000000
- Opcode: 1100 (JMP)
- Base: 010 (R2)
- Assume the value in R2 is 0x3000
- After the execution of the instruction, the value in PC is 0x3000
  - The next instruction to be executed is at address 0x3000

TRAP

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 instruction following TRAP.
(We'll talk about how this works later.)

reviews

- The LC 3 ISA
- The LC 3 instructions and addressing mode
- Understand how an instruction is interpreted by the machine
  - Figure out the meaning of an instruction given in machine code