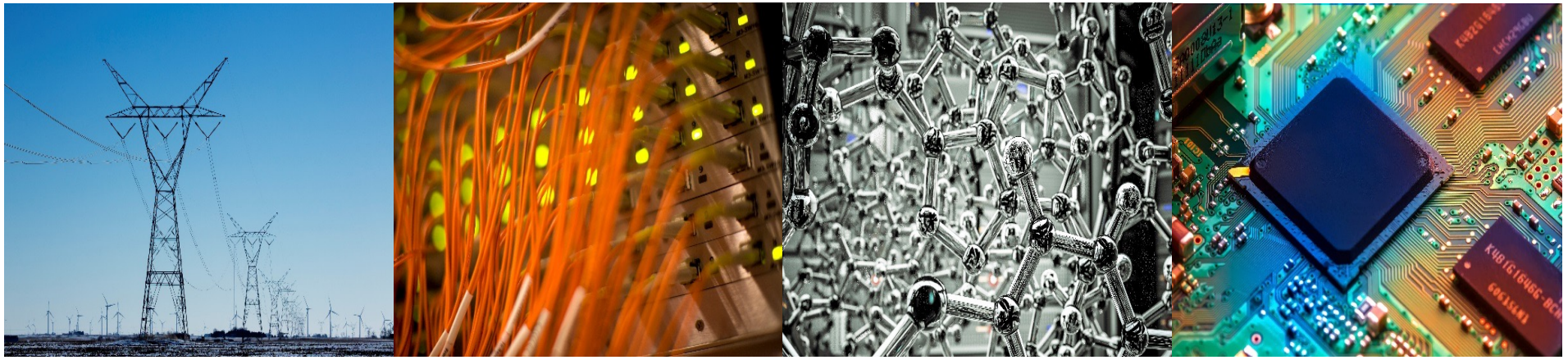
ECE 220 Computer Systems & Programming

Lecture 1 – Memory-Mapped I/O

January 20, 2026



Prof. Yuting W. Chen
Email: ywchen@illinois.edu
Office Hours: 2pm on Tuesdays in ECEB 3064
Course website: courses.grainger.illinois.edu/ece220/sp2026

ILLINOIS
Electrical & Computer Engineering
GRAINGER COLLEGE OF ENGINEERING

Course Logistics

- Lecture BL2: T/R, 11:am to 12:20pm CT, ECEB 1002 (except for exam days)
- Discussion Section (Labs) on Fridays (10 makeup pts/lab worksheet towards MPs, except MP12)
- MPs: due every Thursday @ 10pm CT (100 pts each, late penalty 2pts/hour), 12 MPs, lowest score from MPs (except MP12) will be dropped
- Quizzes: 6 programming quizzes, lowest score dropped (CBTF, in-person)
- Exams: 2 midterms and a final Exam (paper format, in-person)
- Textbook: Patt & Patel, **Introduction to Computing Systems: from bits to gates to C/C++ and beyond**, 2nd or 3rd Edition.
- DRES Accommodation: list Prof. Ivan Abraham as the instructor for TAC, submit LOA to CBTF
- Religious Observance: upload necessary documents as instructed on the course website
- Academic Integrity (FAIR cases)

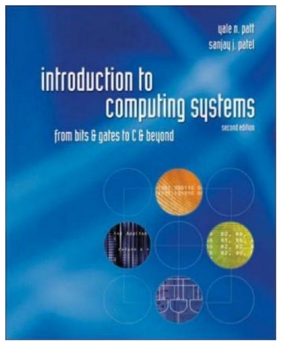
Grading Mechanics:

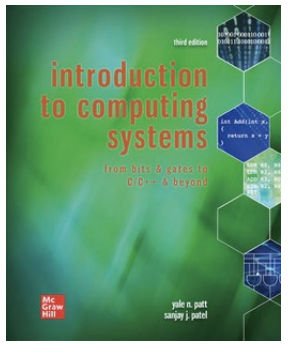
MPs: 15%

CBTF Quizzes: 20%

Midterms: 20% x 2

Final Exam: 25%





Tools & Resources

- **Course website:** all relevant course info, MP write-up, etc.
- **Canvas**: gradebook
- **Github**: MP/LAB code release and submission (**individual**)
- **Gradescope**: lab worksheets (extra-credit) and MP regrades
- **EdStem**: discussion board monitored by TAs
- **CBTF** (PrairieTest): quizzes reservation tool
- **PrairieLearn**: practice quizzes and CBTF testing platform
- **Resources**: CARE, counseling center, DRES

Tips for Academic Success in ECE 220

Keep up with the pace

Do the MPs yourself

Manage your time well

LC-3 Review – ISA (Instruction Set Architecture)

Instruction Set

Data Types: 16-bit 2's complement integers

Addressing Modes (how the location of operand is specified):

- Non-memory addresses – immediate (part of instruction), register
- Memory address – PC-relative, base+offset, indirect

Opcodes (16-bit, bits 12-15 used to specify the opcode):

- *Operate* instructions: ADD, AND, NOT
- *Data movement* instructions: LD, LDI, LDR, LEA, ST, STR, STI
- *Control* instructions: BR, JSR/JSRR, JMP, RET, TRAP, RTI
- Condition codes: N (negative), Z (zero), P (positive)

LC-3 Review – Using LD, LDI, LDR, LEA

```
.ORIG x3000
LD      R6, LABEL
LDI     R6, LABEL
LDR     R2, R6, #1
LEA     R2, LABEL
LABEL   .FILL x5001
.END

; Assume the following
; Address   Data
; x5001     x6001
; …         …
; x6001     x7001
; x6002     x7002
```

LC-3 Warm-up

1. Initialize a register to zero

2. Copy value from one register to another

3. Compute 8 - 2

4. Compute 6 x 4

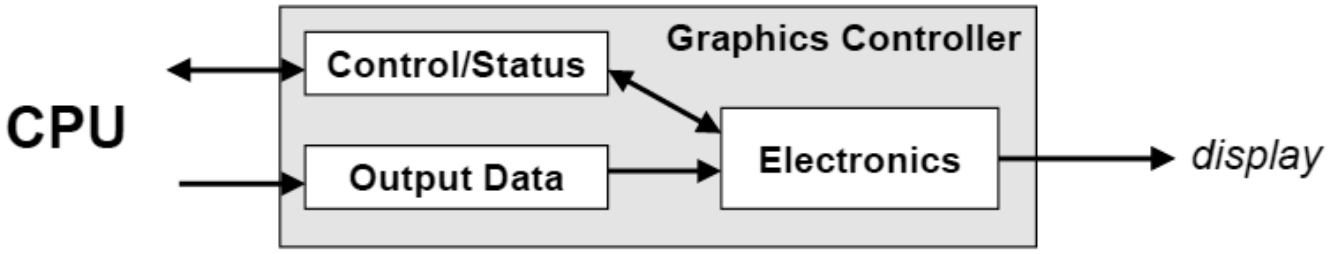
I/O and Basics of Interface Design

I/O is for interfacing the physical world and the digital world.

- Producer of data (finger at touchscreen) is working much much more slowly than consumer of that data (messaging app)
- We need to account for ***asynchronous*** operation
- We will use a simple consumer/producer handshake

➢ Can you think of other I/O devices that you use regularly?

I/O Controller



Control/Status Registers

- CPU **tells** device what to do: **write** to control register
- CPU **checks** whether task is done: **read** status register

Data Registers

- CPU transfers data to/from device

Device Electronics

- Performs actual operation (pixels to screen, character from keyboard)

Memory-Mapped I/O

- Assign a memory address to each device register
- Use data movement instructions (load/store) for control and data transfer

LC-3 Input and Output Device Registers

_____ -- store **ASCII value** of character entered from **keyboard**

_____ -- let processor know a new value is entered

_____ -- store **ASCII value** of character to be displayed on **monitor**

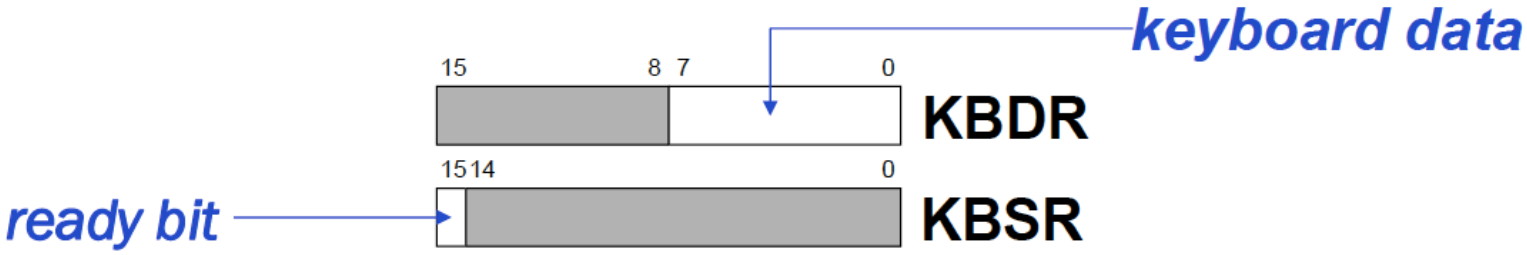
_____ -- let processor know a new value is ready to be displayed

LC-3 Memory-Mapped Device Registers

Address	Contents	Comments
x0000		; system space
…		
x3000		; user space
		; programs
		; and data
…		
xFE00	KBSR	; Device register
xFE02	KBDR	
xFE04	DSR	
xFE06	DDR	
…		
xFFFF		

- These are the memory addresses to which the device registers (KBDR, etc.) are mapped
- But the device registers physically are separate from the memory
- Memory-mapping device registers is a very common way to design interfaces for computing systems

Read from the Keyboard – Handshaking Using KBDR/KBSR



❖ ***KBSR[15] controls the synchronization*** *of the slow keyboard and the fast processor.*

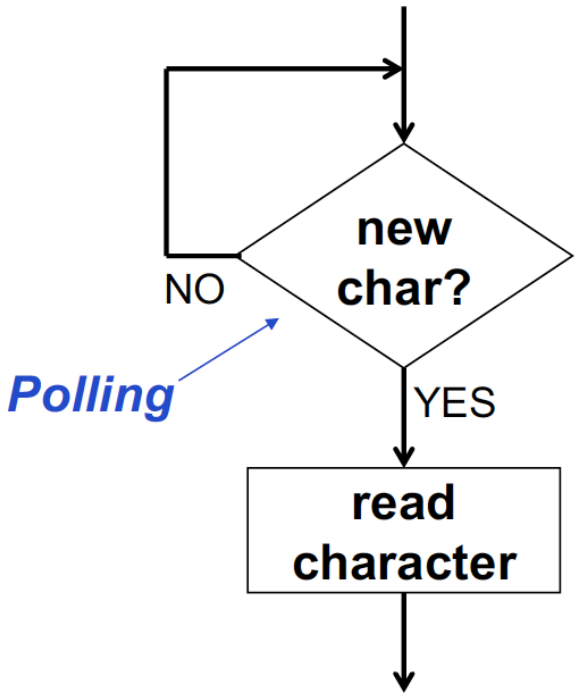
When a key on the keyboard is struck

- _____ is placed in KBDR[7:0] (upper bits are zero)
- _____ (KBSR[15]) is automatically set to 1 (by keyboard electronic circuit)
- Keyboard is _____ (new character entries will be ignored)

When KBDR is read

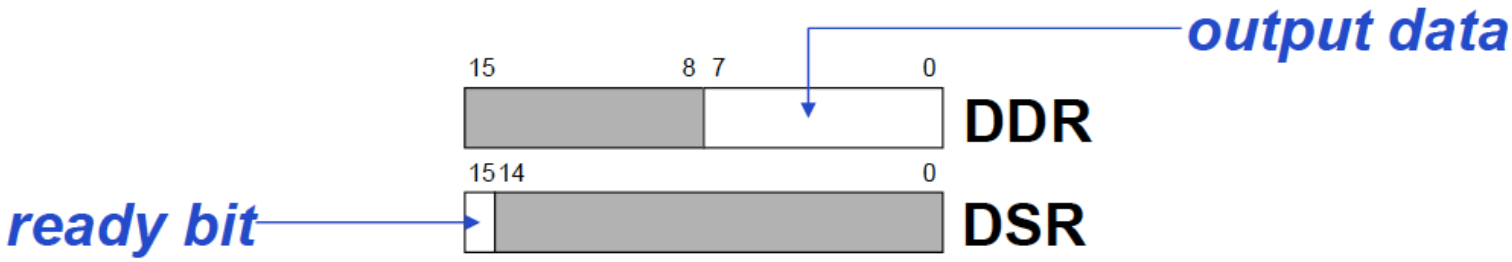
- _____ is automatically set to 0 (by keyboard electronic circuit)
- Keyboard is _____

Read from the Keyboard – Basic LC-3 Routine




```
.ORIG x3000

;set up a loop to check ready bit in KBSR



;branch to the beginning if there is no KB input



;otherwise, load data from KBDR to R0



HALT
KBSR_ADDR   .FILL     xFE00
KBDR_ADDR   .FILL     xFE02
.END
```

Output to the Monitor – Handshaking Using DDR/DSR



❖ ***DSR[15] controls the synchronization*** *of the fast processor and slow monitor display.*

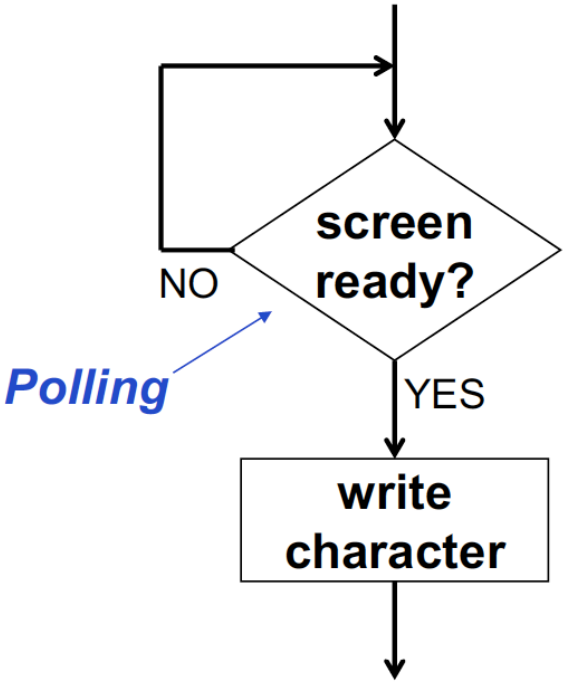
When monitor is ready to display a new character

- _____ (DSR[15]) is automatically set to 1 (by monitor electronic circuits)

When data is written to DDR

- _____ is automatically set to 0 (by monitor electronic circuits) and character in _____ is displayed
- Any other character data written to DDR is _____ while DSR[___] is zero.

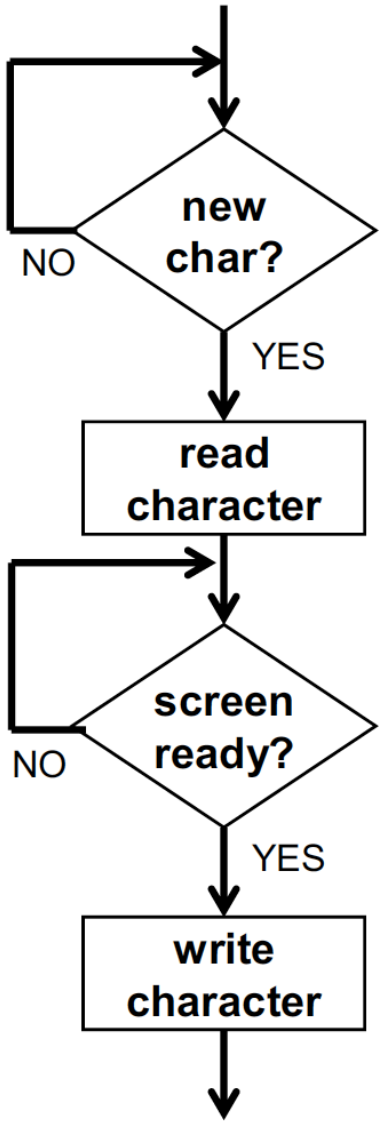
Output to the Monitor – Basic LC-3 Routine




```
.ORIG x3000

;set up a loop to check ready bit in DSR



;branch to the beginning if display is
;not ready for new data



;otherwise, store data from R0 to DDR



HALT
DSR_ADDR    .FILL     xFE04
DDR_ADDR    .FILL     xFE06
.END
```

Echo Routine Implementation

```
.ORIG x3000
```




```
HALT
KBSR_ADDR   .FILL xFE00
KBDR_ADDR   .FILL xFE02
DSR_ADDR    .FILL xFE04
DDR_ADDR    .FILL xFE06
.END
```