University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

# ECE 220: Computer Systems and Programming

## Lecture 1: Input/Output Abstractions

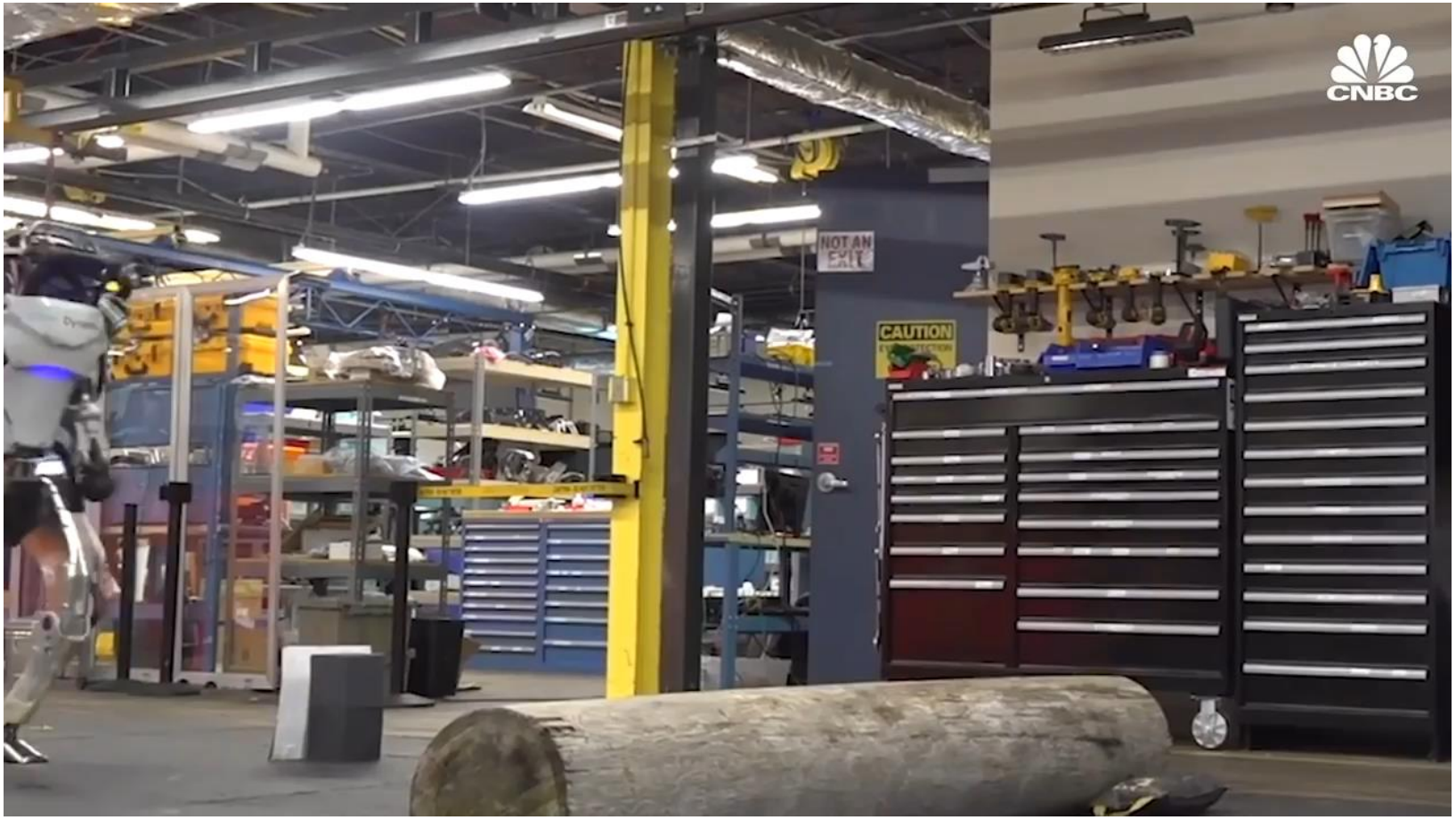
Instructor: Ujjal Kumar Bhowmik
Section: BL, 12:30-1:50PM, ECEB 1002
Office Hours: 3-4PM, Tuesday, ECEB 2022
Course Website:
https://courses.grainger.illinois.edu/ece220/sp2024/

# Outline

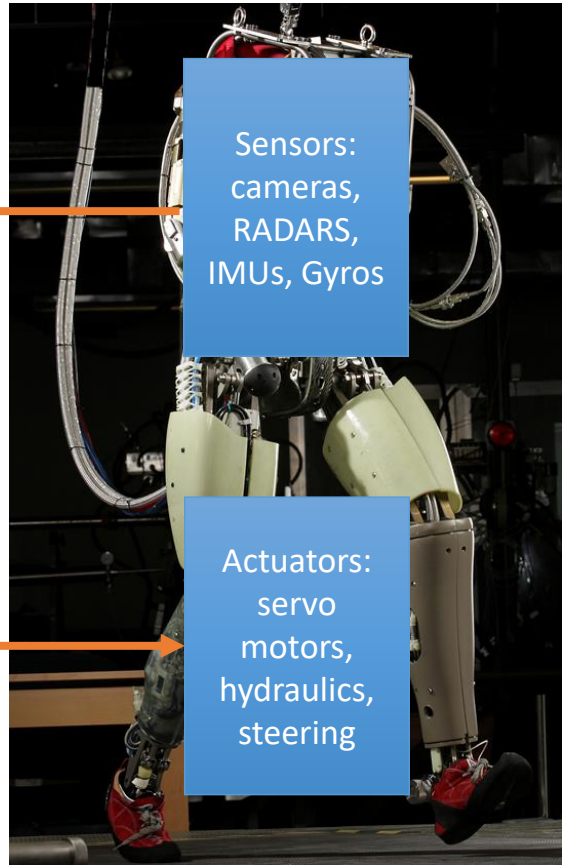
- Section 9.2 (3rd Ed.), 8.1-8.4 (2nd Ed.) of Patt and Patel
- I/O principles
- Input from keyboard
- Output to monitor

- Key concepts
  - Memory mapped I/O
  - Asynchronous and synchronous communication

# Humanoid Robot

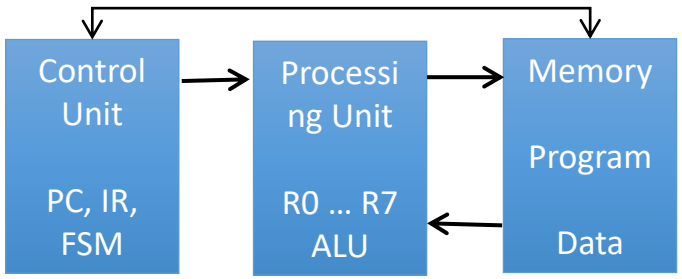


# I/O with the physical world




CPU & Memory
Controlling bipedal gait, Image processing, obstacle detecting, path planning, control, …

CPU and Memory



# I/O and Basics of Interface Design

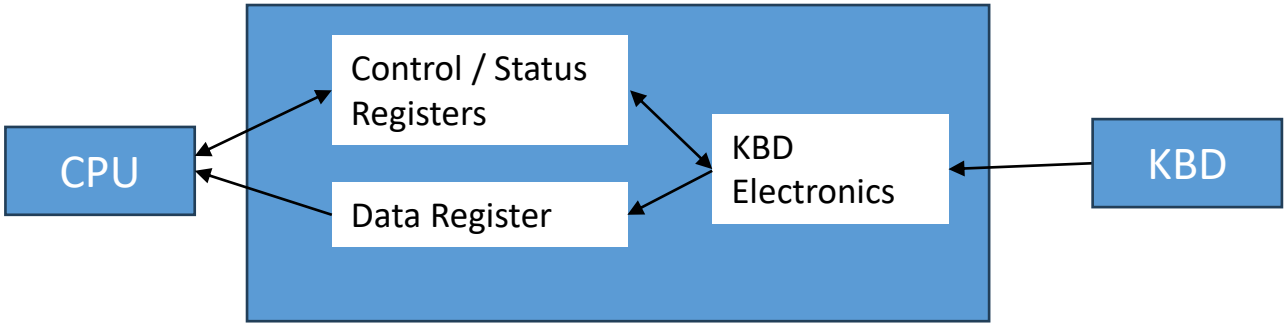
I/O is for interfacing the physical world and the digital world.

- Producer of data (sensors) is working much more slowly than consumer of that data (processor/program)
- We need to account for ***asynchronous*** operation
- We will use a simple consumer/producer handshake

For LC3 we just need to consider…. Input/output?

# I/O Device Controller

## Keyboard Interfacing:




**Control/Status Register:**

CPU tells the device what to do: **write** to control register

CPU checks whether a new key is pressed: **read** the status register

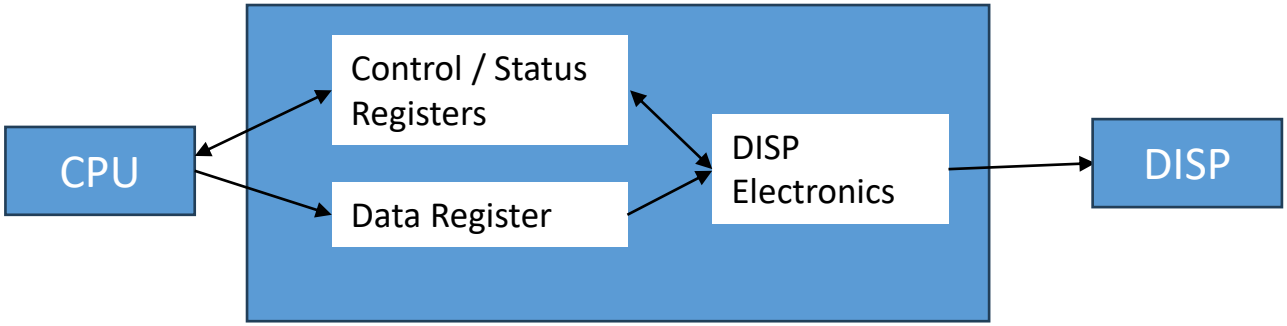
**Data Register:**

CPU reads the ASCII value of the key pressed

**KBD Electronics**: Performs actual operation (character from keyboard)

# I/O Device Controller

## Display Interfacing:




**Control/Status Register:**

CPU tells the device what to do: **write** to control register

CPU checks whether a last character is displayed: **read** the status register

**Data Register:**

CPU sends the ASCII value of the character to be displayed

**DISP Electronics**: Performs actual operation (character to the screen)

# Memory-Mapped I/O

- **Assign a memory address to each device register**
  - I/O device registers are mapped to set of addresses that are allocated to I/O device registers rather than to memory locations.
- **Use data movement instructions (load/store) for control and data transfer**

**LC-3 Input and Output Device Registers**

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

**KBSR** - let processor know a new value is entered

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

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

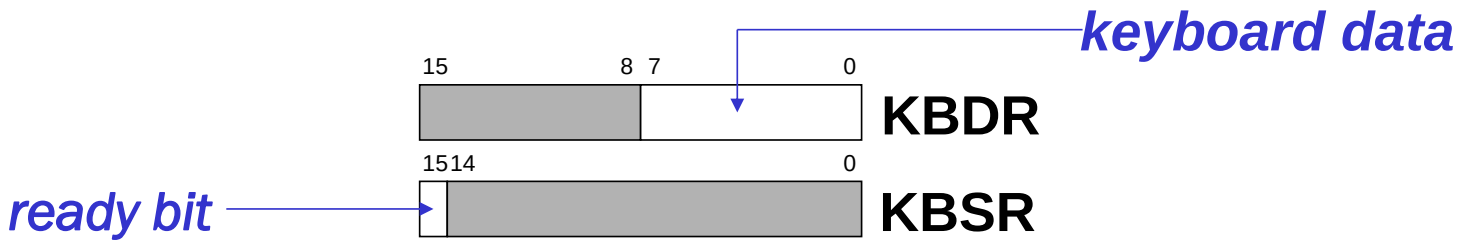
# LC3 Memory: Memory mapped device registers

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

These are the memory addresses to which the device registers (KBDR, etc.) are mapped

The device registers physically are separate circuits from the memory

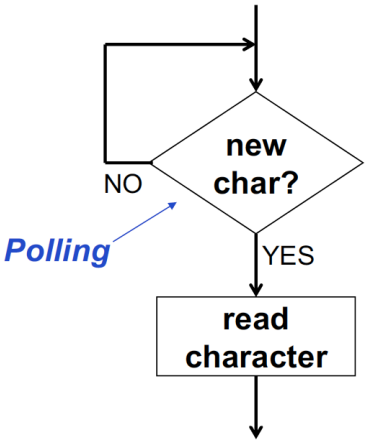
# Handshaking using KBDR and KBSR




- When a char is typed by user in the keyboard
  - Its ASCII code is placed in KBDR[0:7]
  - KBSR[15] is set to 1 (ready bit)
  - Keyboard is disabled, i.e., any further keypress is ignored
- When KBDR is read by CPU
  - KBSR[15] is set to 0
  - Keyboard is enabled

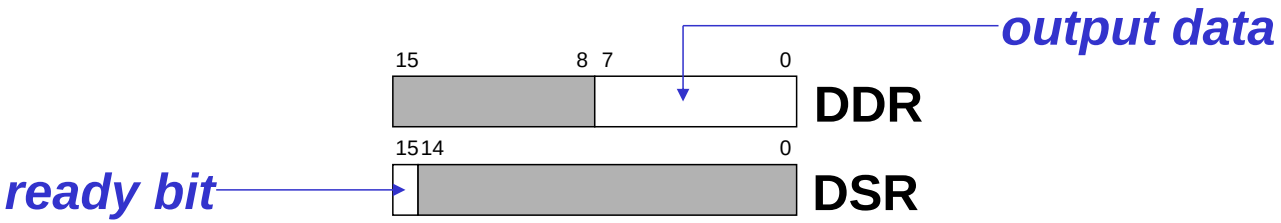
This is part of the keyboard Hardware.

# LC-3 Basic Instructions to Read from the Keyboard




```
.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
```

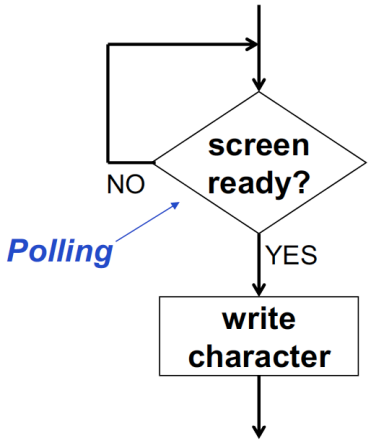
# Handshaking using DDR and DSR




- When monitor is ready to display another char
  - DSR[15] is set to 1: (**ready bit**)
- When new char is written to DDR
  - DSR[15] is set to 0
  - Any other chars written to DDR are ignored
  - DDR[7:0] is displayed

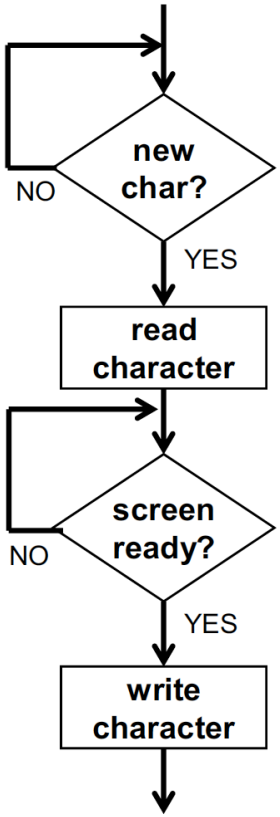
This is part of the display hardware.

# Use LC3 LOAD/STORE Instructions to Display a Character to the Monitor




```
.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
.EN
```

# Write code for ECHO (read a char and display it)



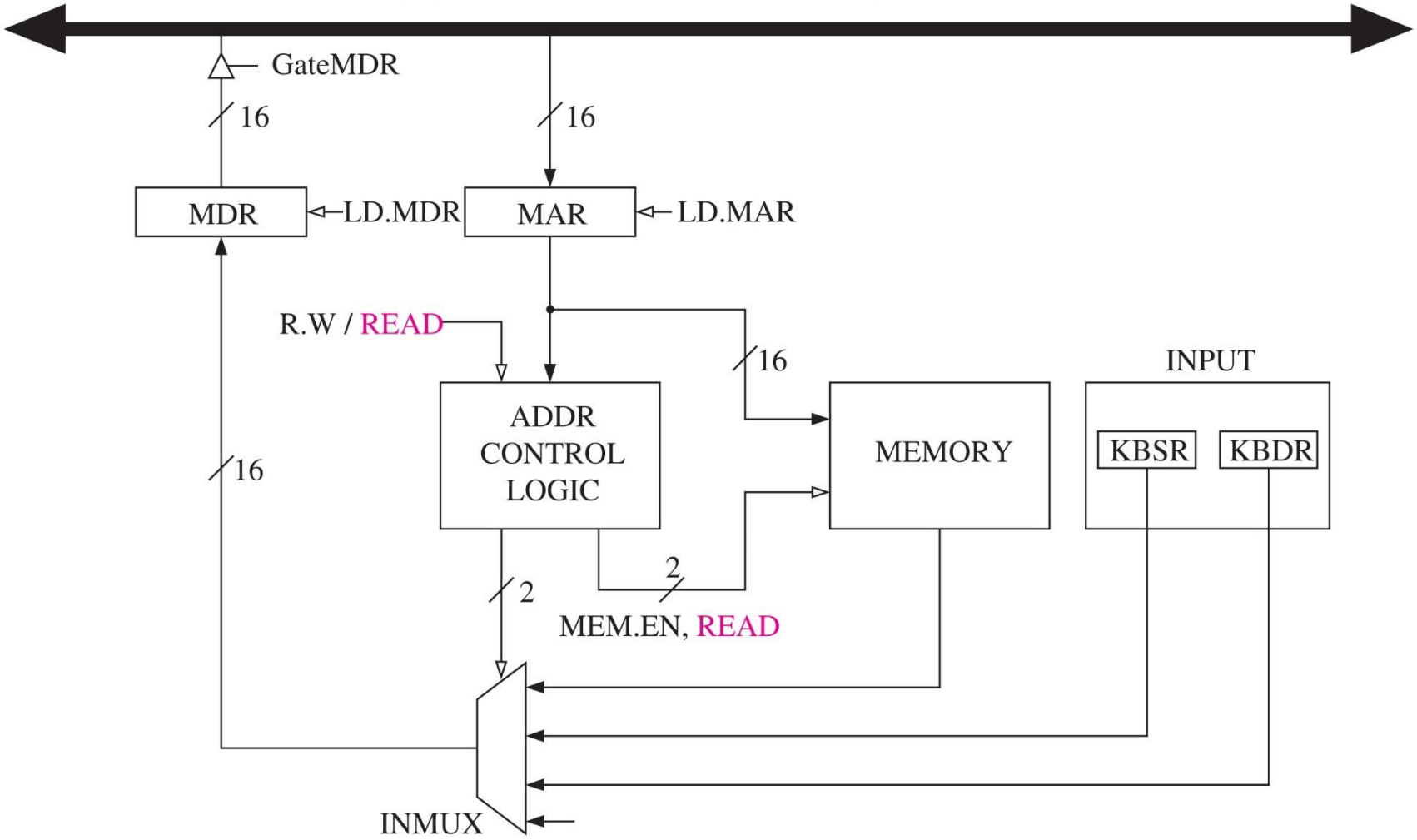

```
.ORIG x3000


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

# What does this code do?

```
.ORIG x3000

KPOLL   LDI R0, KBSR _ADDR
        BRzp KPOLL
        LDI R0, KBDR _ADDR
DPOLL   LDI R1, DSR _ADDR
        BRzp DPOLL
        STI R0, DDR _ADDR
HALT

KBSR_ADDR .FILL xFE00
KBDR_ADDR .FILL xFE02
DSR _ADDR  .FILL xFE04
DDR _ADDR  .FILL xFE06
.END
```

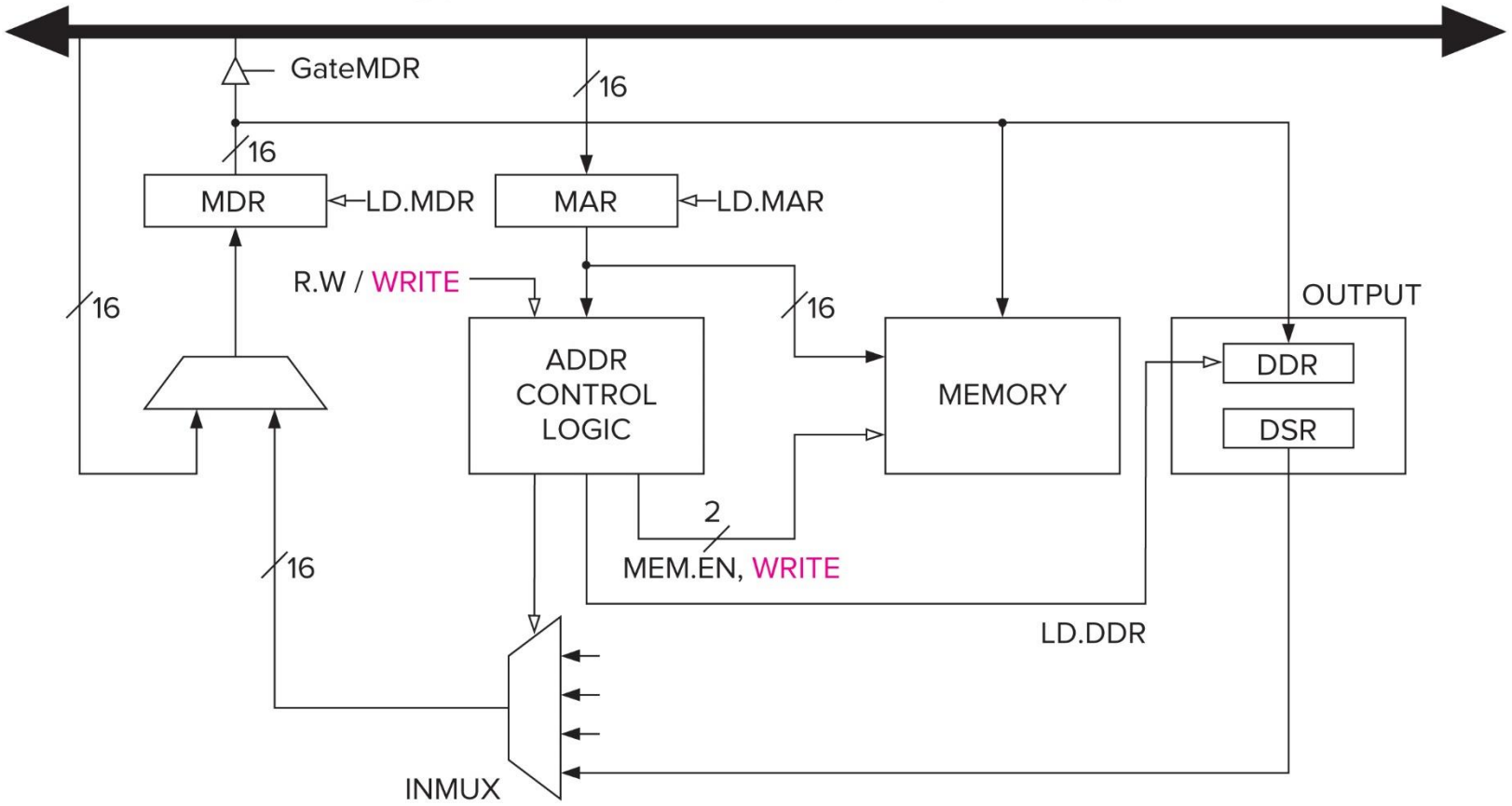
# Simplified Memory-Mapped Input





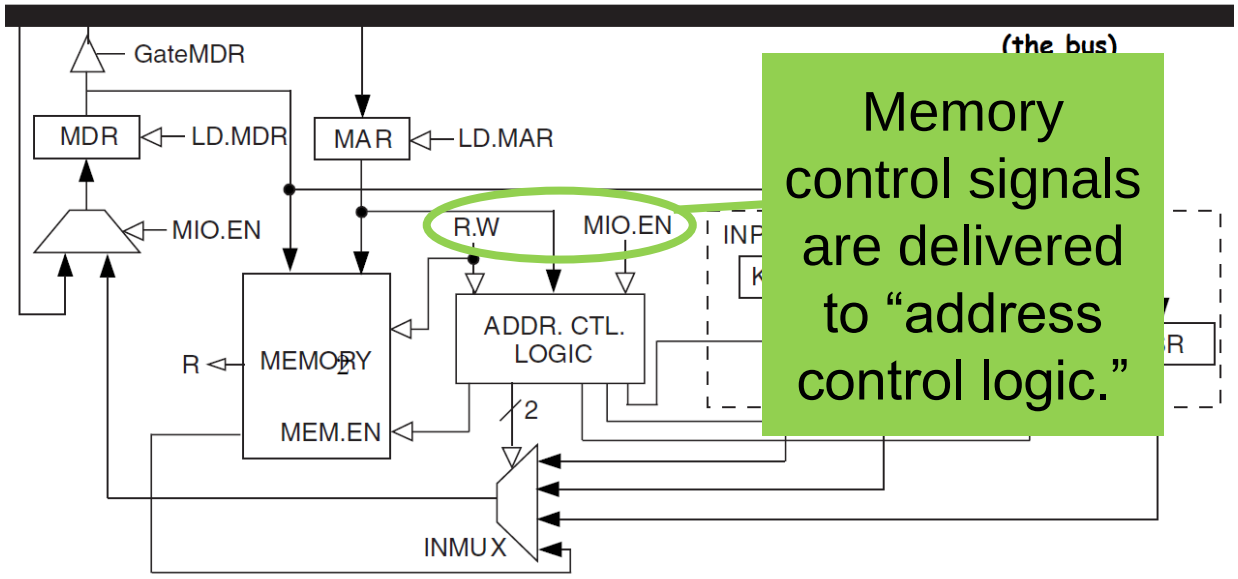
# Simplified Memory-Mapped Output (monitor)



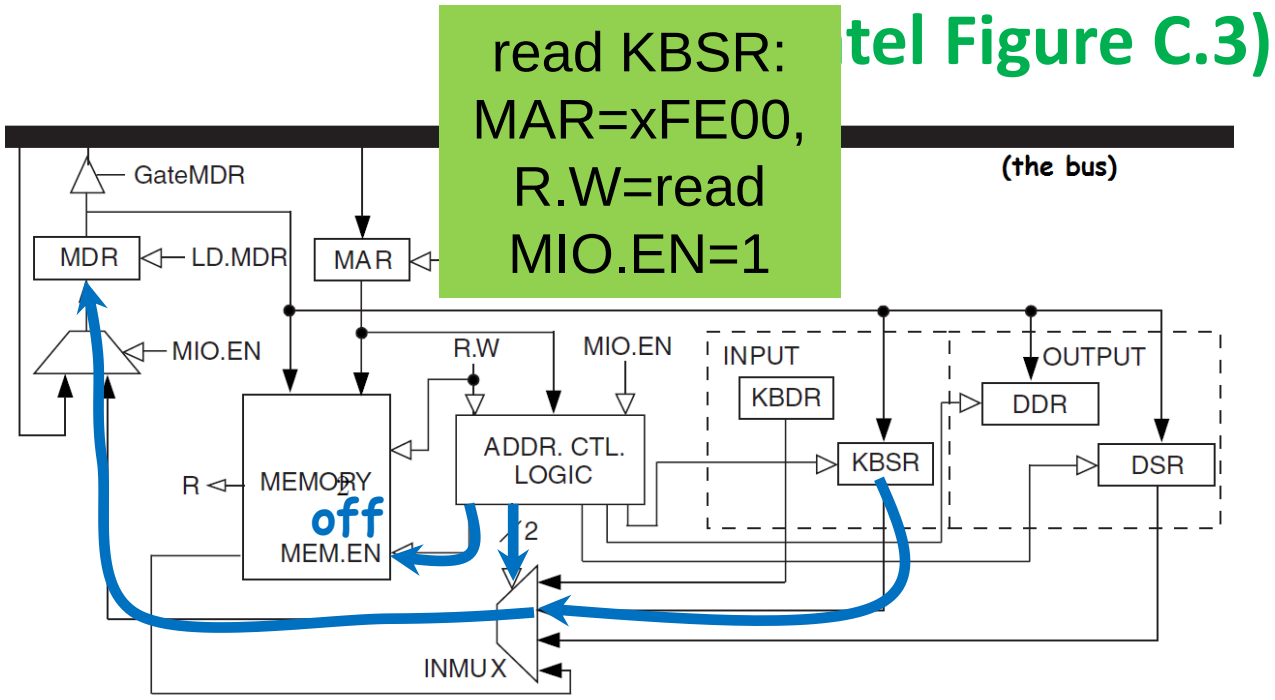


# P&P Appendix C Describes I/O Memory Mapping

## (Patt and Patel Figure C.3)

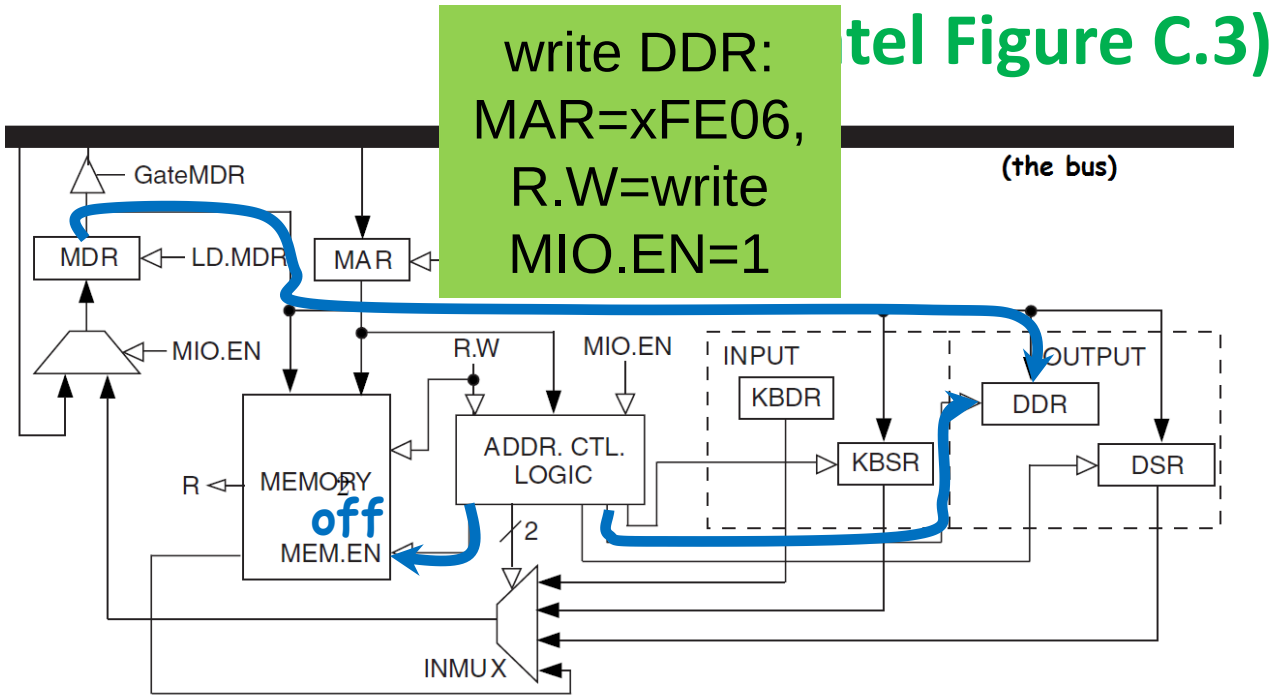


# Example: Reading the KBSR



# Example: Writing the DDR

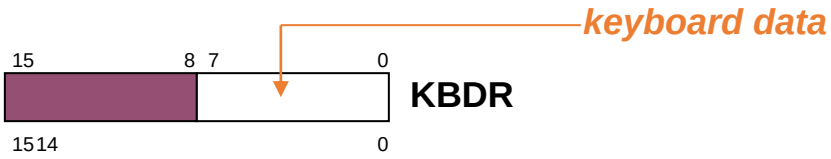
write DDR:
MAR=xFE06,
R.W=write
MIO.EN=1



# Exercises

- Write code for ECHO (read a char and display it)
- Write code for PUTS (display a stored string)
- Write a more sophisticated input function using command prompt.

# Reading Input (first attempt)

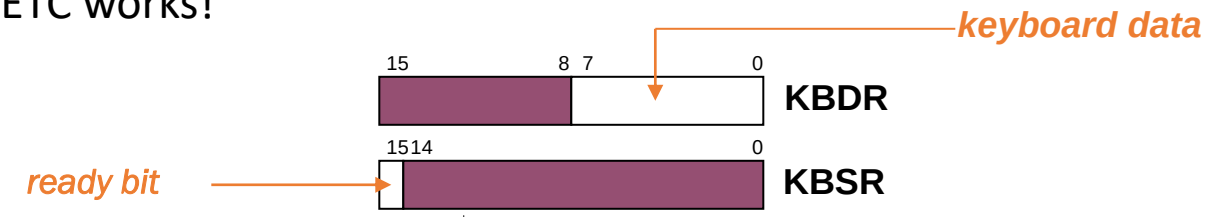



```
START           LDI     R1, KBDRAdd     ; Read from KBD
                ….
                BRnzp   START
KBDRAdd         .FILL     xFE02         ; Address of KBDR
```

Does this work?

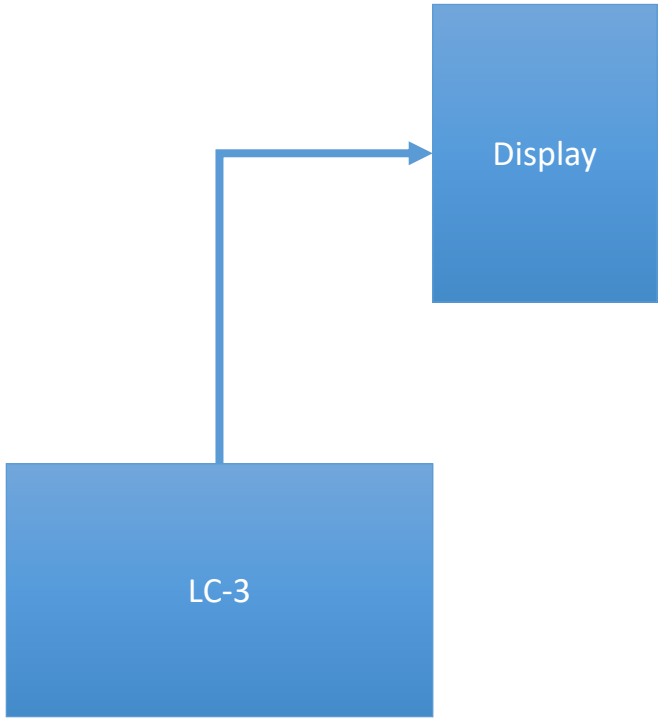
# Reading Input the right way

This is how TRAP x20 = GETC works!




```
START           LDI     R1, KBSR_ADDR   ; Test for
                BRzp    START           ; character input
                LDI     R0, KBDR_ADDR
                BRnzp   NEXT_TASK       ; Go to the next
task
                ...
KBSR_ADDR       .FILL     xFE00         ; Address of KBSR
KBDR_ADDR       .FILL     xFE02         ; Address of KBDR
```

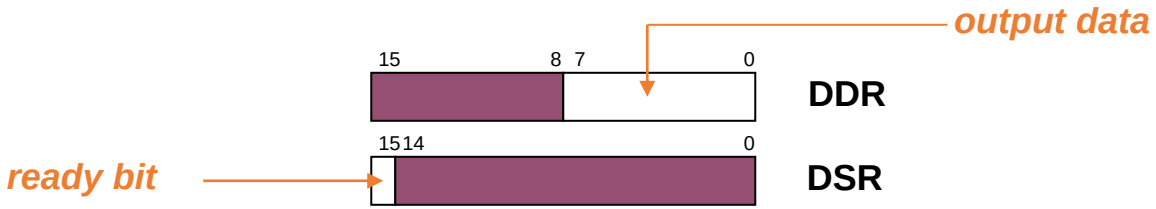
# I/O Layout




- How to connect a display to LC3?

# Writing TRAP x21

This is how TRAP x21 = OUT works!




```
START           LDI     R1, DSR_ADDR    ; Test for
                BRzp    START           ; character input
                STI     R0, DDR_ADDR
                BRnzp   NEXT_TASK       ; Go to the next task
                ...
DSR_ADDR        .FILL     xFE04         ; Address of DSR
DDR_ADDR        .FILL     xFE06         ; Address of DDR
```

# Summary of concepts

- Memory mapped I/O (extra hardware for flexibility and convenience of programming)
- Asynchrony
- Polling