ECE 220 Computer Systems & Programming

Lecture 2: Input/Output Abstractions

August 29, 2019

ECE ILLINOIS

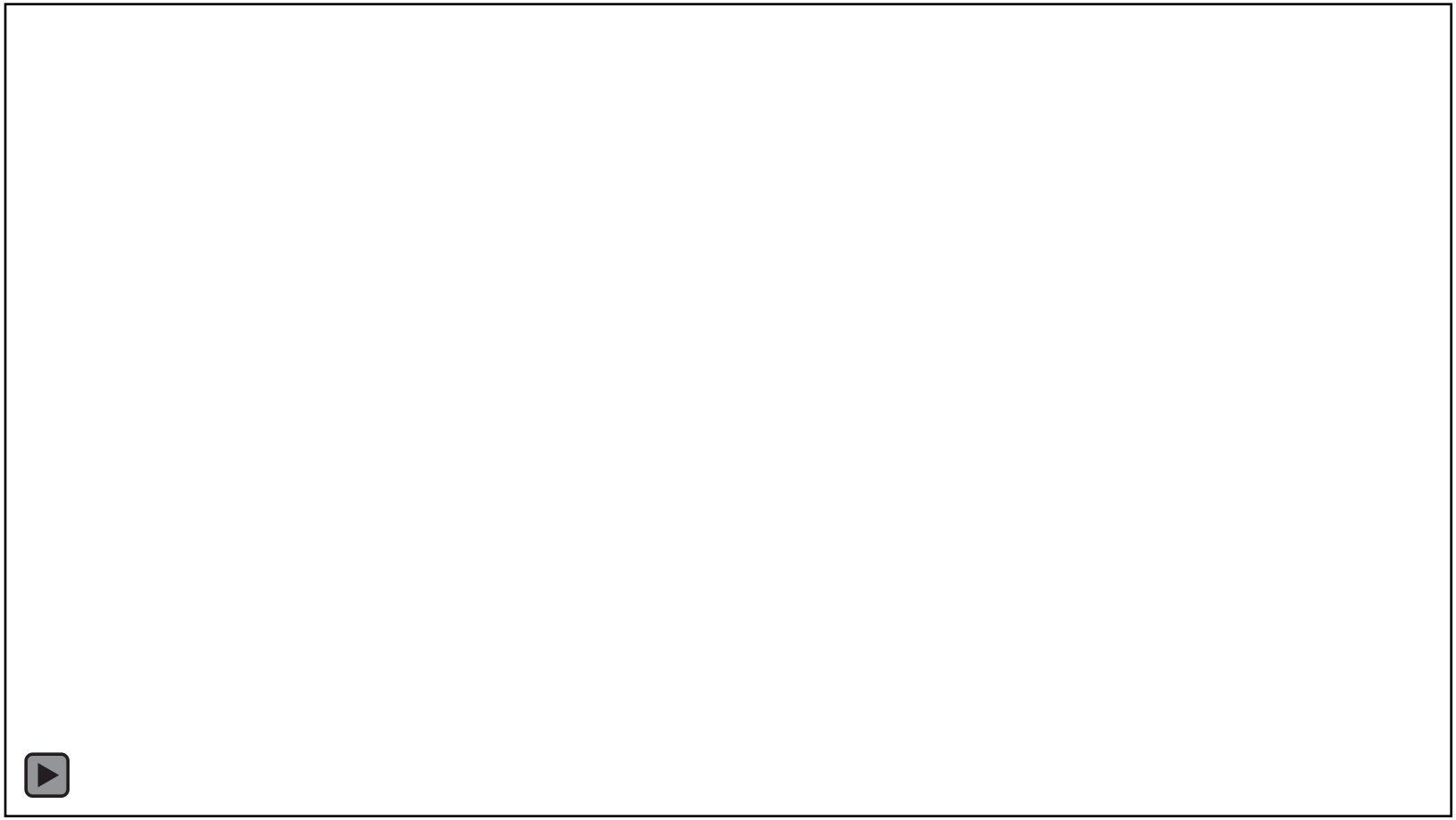
ILLINOIS

Outline

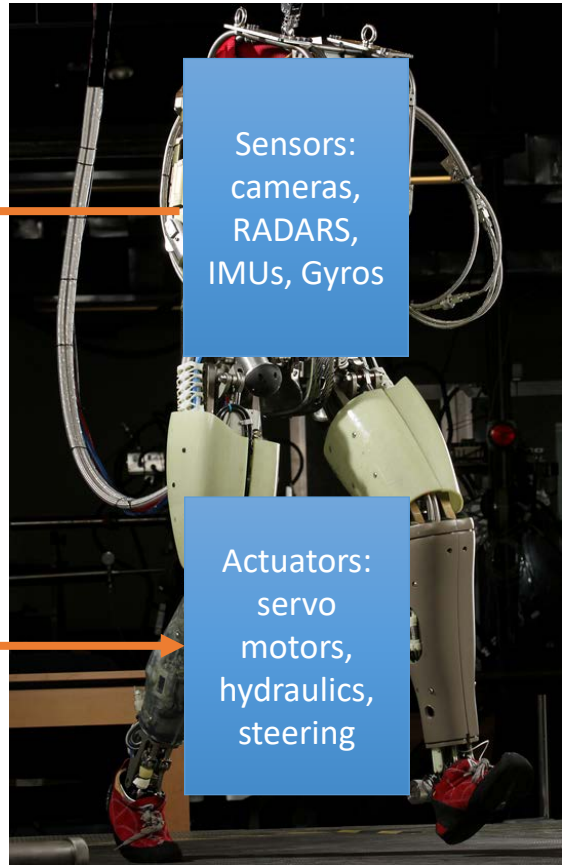
- Section 8.1-8.4 of Patt and Patel
- I/O principles
- Input from keyboard
- Output to monitor (reading assignment)

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

Humanoid Robot

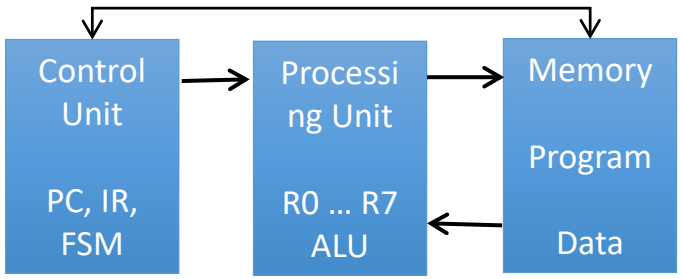


I/O with the physical world

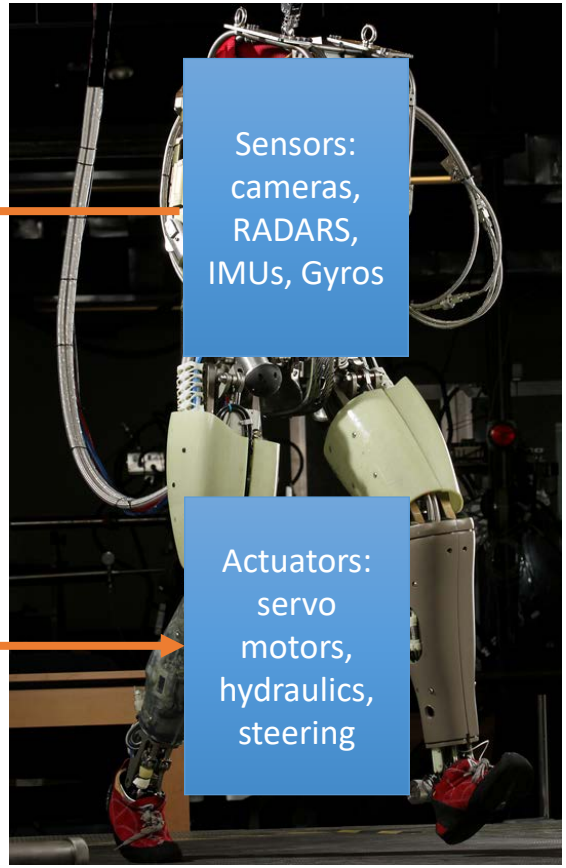


CPU & Memory
Controlling bipedal gait, Image processing, obstacle detecting, path planning, control, …

CPU and Memory

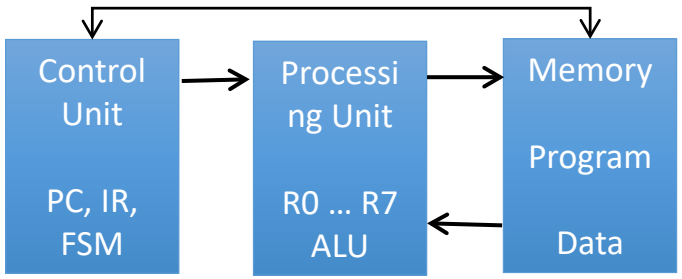


Complete system



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

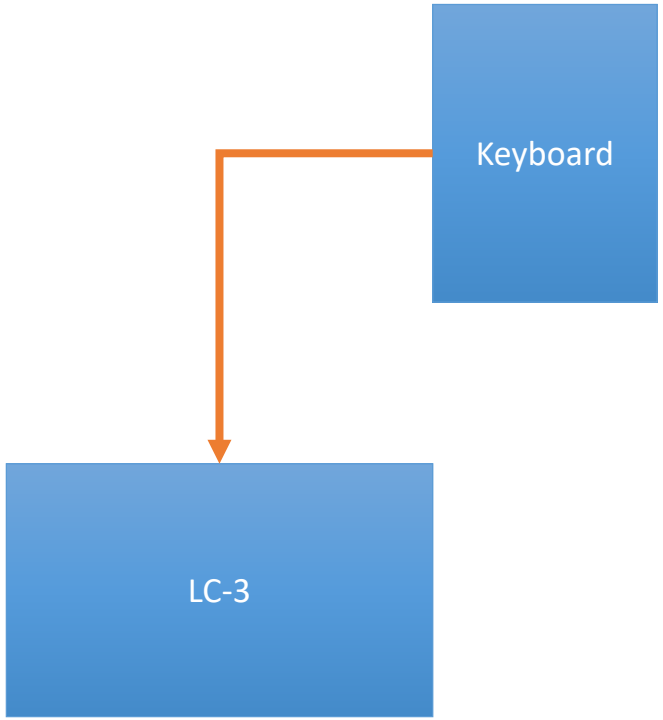
CPU and Memory



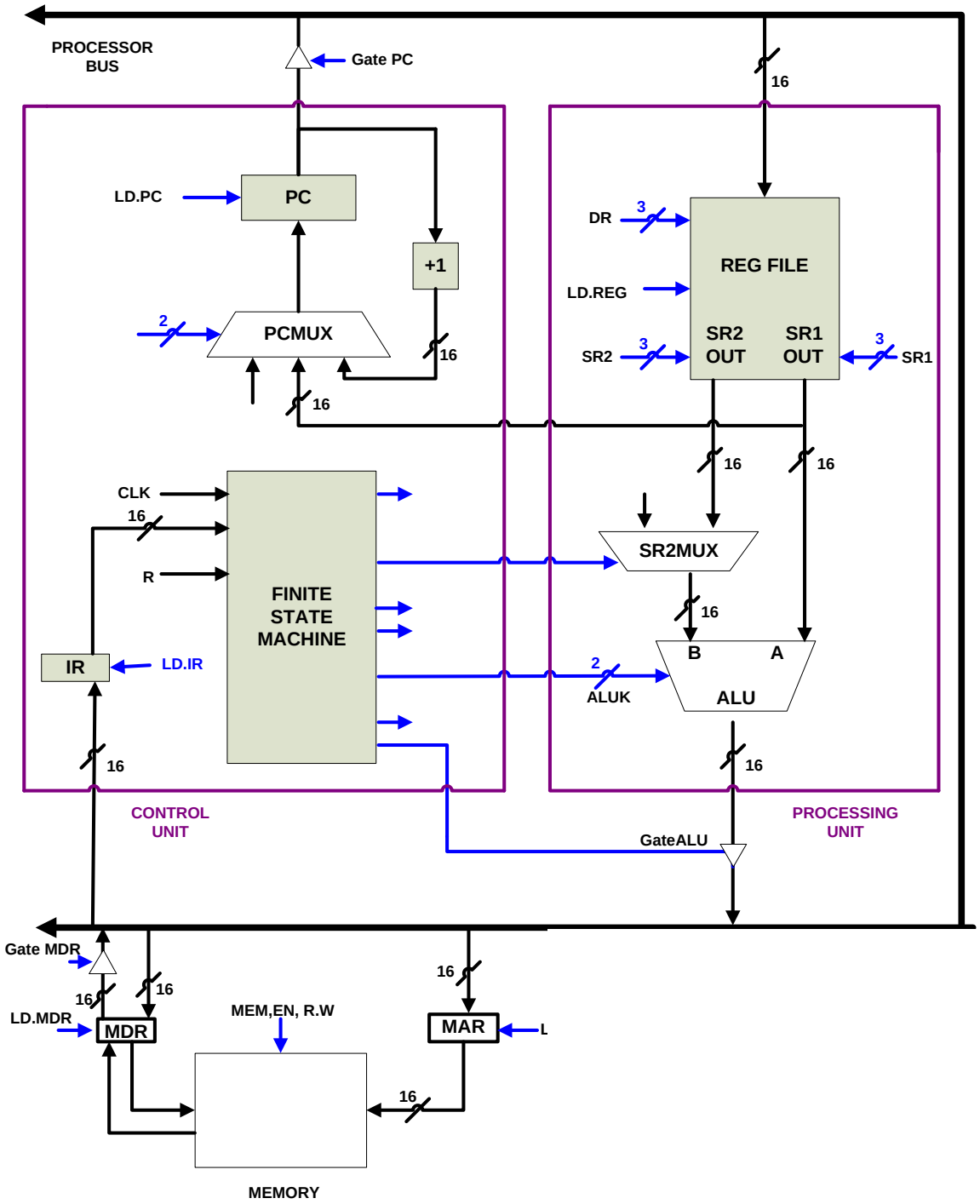
Many more examples

- Autopilot
- Antilock brakes
- …
- Autonomous cars
- UAVs/Drones
- Space rovers
- Smart pacemakers
- Powerplants
- …

I/O Layout



- How to connect a keyboard to LC3?
  - How to synchronize with the processor?
- Interrupt I/O
- Memory Mapped I/O
  - Using existing instruction (LC3)
  - Using IN/OUT separate instructions



how should we connect a keyboard to the computer?

LC3 Memory: Memory mapped device registers

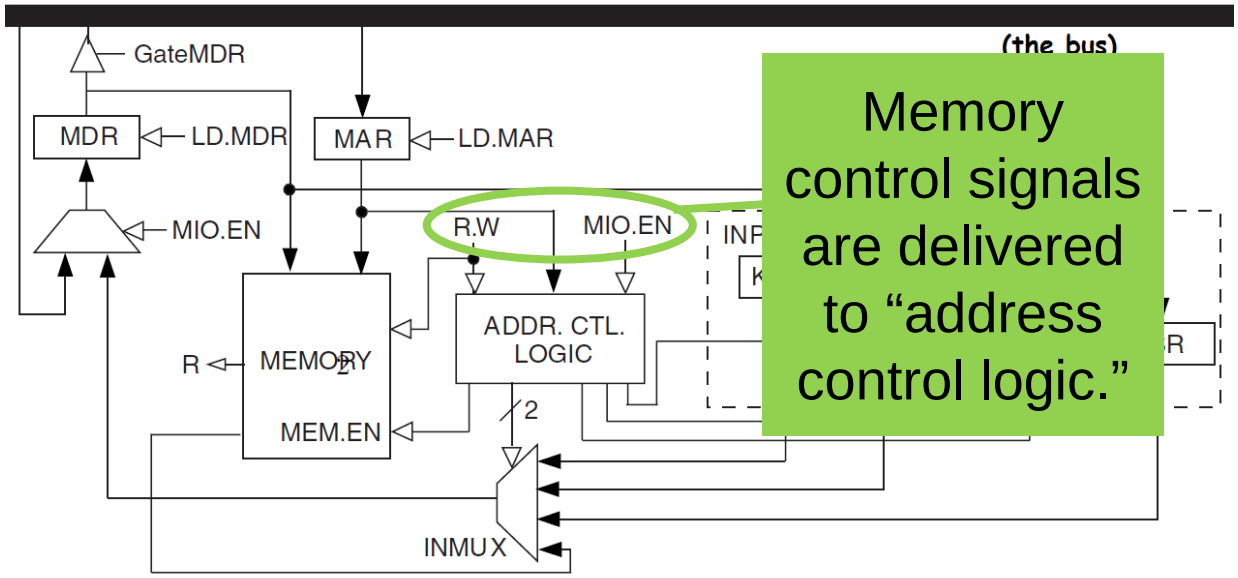
Address	Contents	Comments
x0000		;system space
…		
x3000		; user space
		; programs
		; and data
…		
xFE00	KBSR	; Device registers maps
xFE02	KBDR	
xFE04	DSR	
xFE06	DDR	
…		
xFFFF		

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

The device registers physically are separate circuits from the memory

P&P Appendix C Describes I/O Memory Mapping

(Patt and Patel Figure C.3)

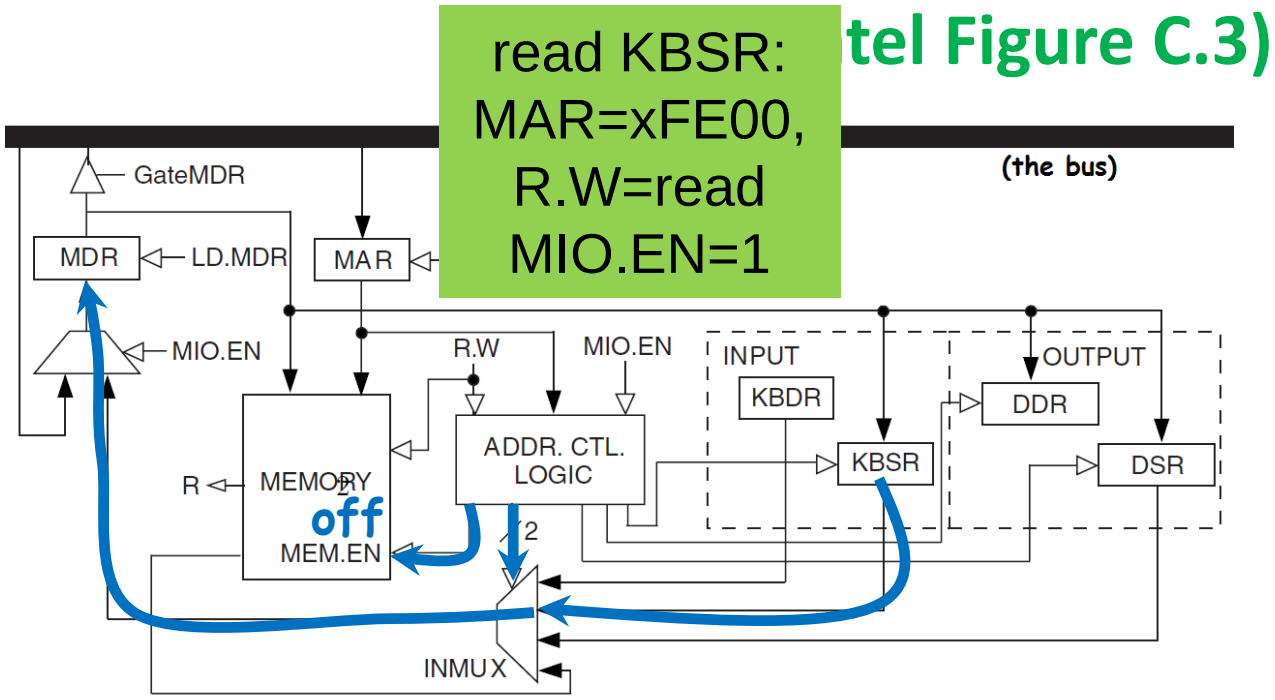


Keyboard and Display are Memory-Mapped in LC-3

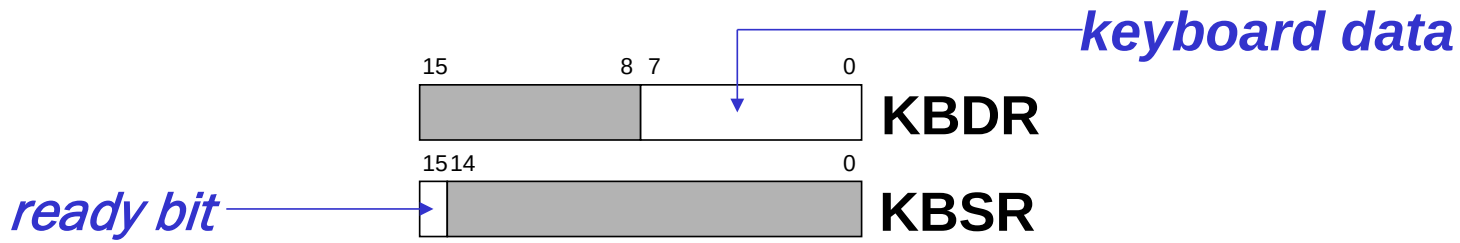
- Based on MAR and these control signals, the **address control logic controls**:
  - **memory enable** (chip select) signal actually delivered to the memory,
  - **load control for DDR**,* and
  - **INMUX select** lines, which determines whether memory, KBSR, KBDR, or DSR writes the load result to MDR.

*And for KBSR and DSR, but we'll explain why later.

Example: Reading the KBSR



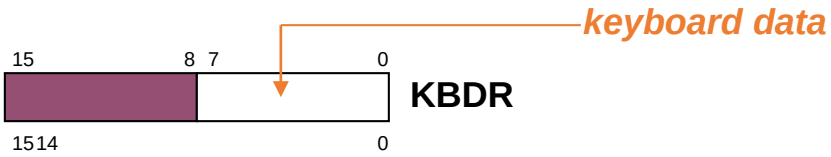
Handshaking using KBDR and KBSR



- 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.

Reading Input (first attempt)

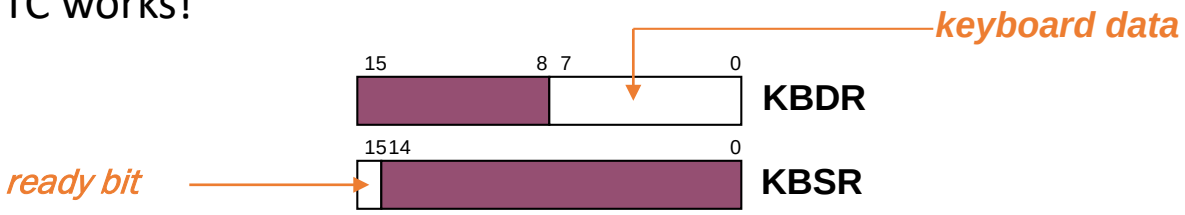



```
START           LDI     R1, KBDRAdd     ; Read from KBD
                ….
                BRnzp   START
KBDRAdd         .FILL      xFE02        ; Address of KBDR
```

Does this work?

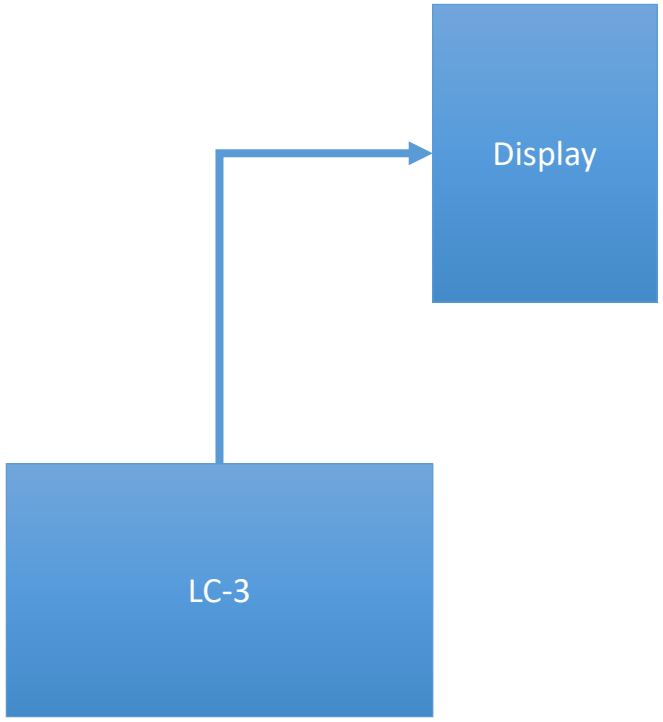
Reading Input the right way

This is how TRAP x20 = GETC works!



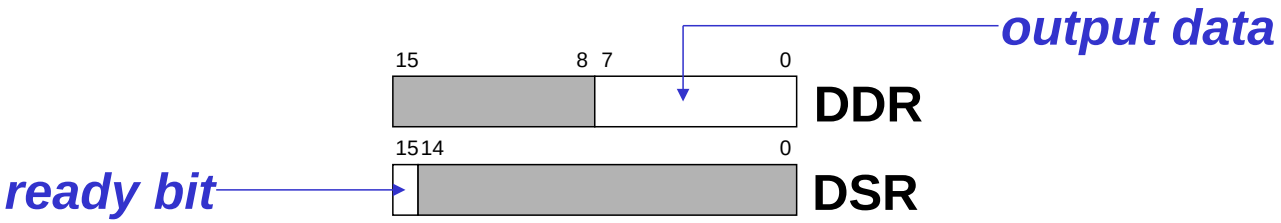

```
START           LDI     R1, KBSR_ADDR   ; Test for
                BRzp    START           ; character input
                LDI     R0, KBDR_ADDR
                BRnzp   NEXT_TASK       ; Go to the next
task
                ...
KBSR_ADDR       .FILL      xFE00        ; Address of KBSR
KBDR_ADDR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

I/O Layout



- How to connect a display to LC3?

Handshaking using DDR and DSR



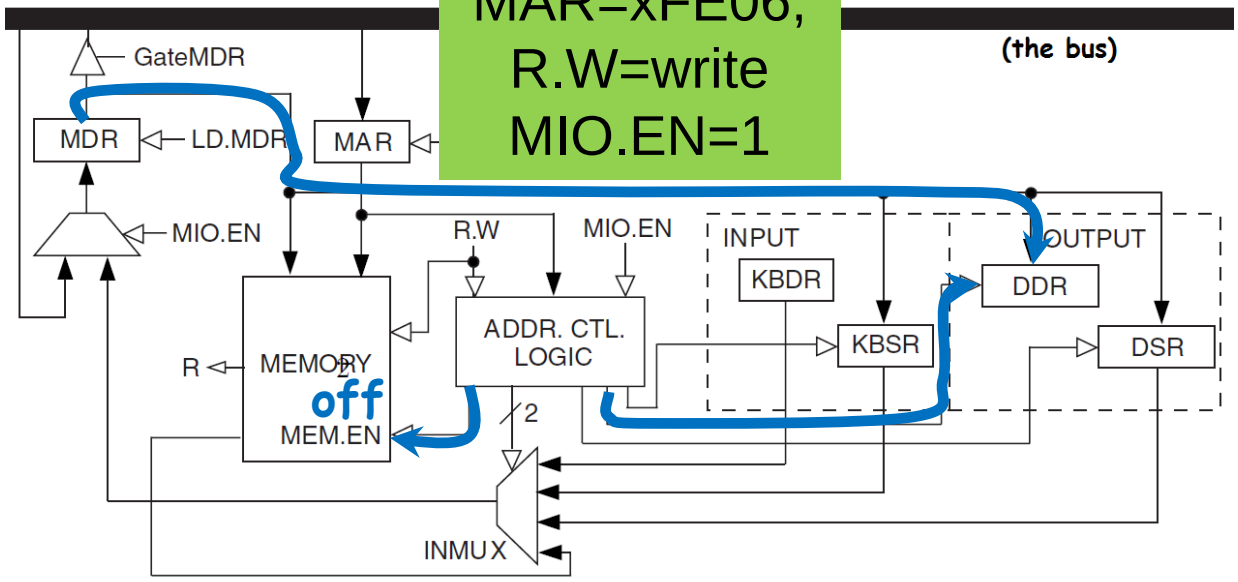
- When monitor is ready to display another char
  - DSR[15] is set to 1: (**ready bit**)
- When new char is written to DDR
  - DSR[15] is set to 0
  - Any other chars written to DDR are ignored
  - DDR[7:0] is displayed

This is part of the display hardware.

Example: Writing the DDR

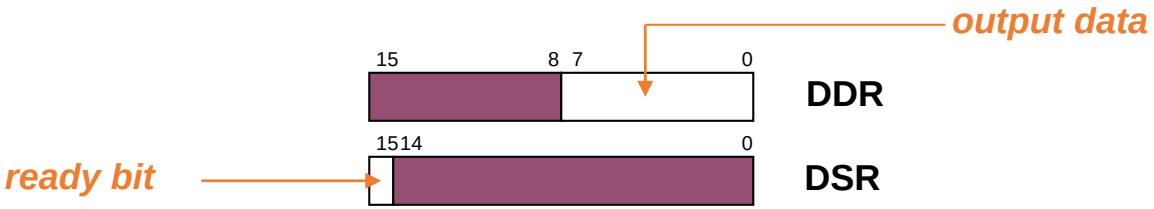
write DDR:
MAR=xFE06,
R.W=write
MIO.EN=1

tel Figure C.3)



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
```

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.

- Read Interrupt-driven I/O

Summary of concepts

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