ECE 220
Lecture 02: Input/Output Abstractions

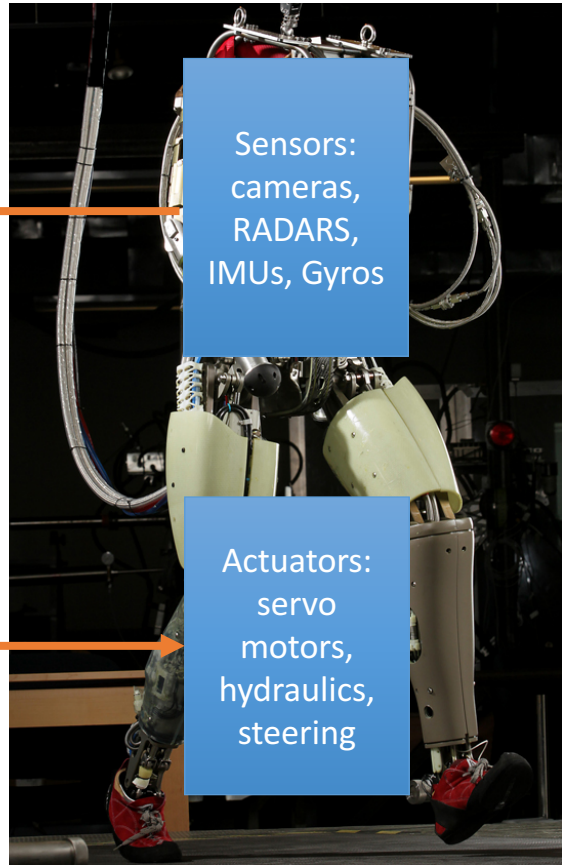
Slides by Prof. Sayan Mitra

Outline

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

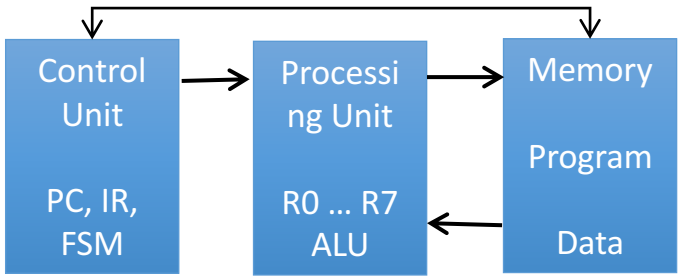
- Key concepts
  - Memory mapped I/O
  - Asynchronous and synchronous communication

I/O with the physical world

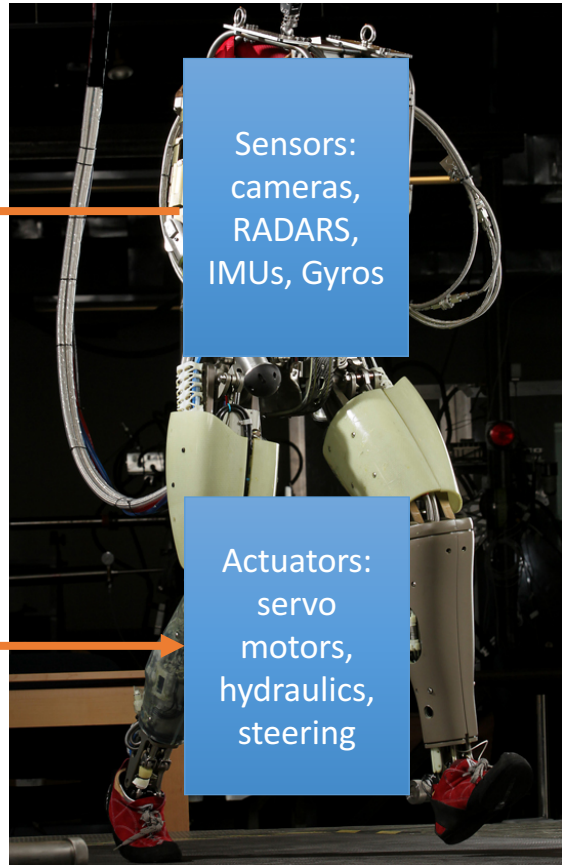


CPU & Memory
Controlling bipedal gait, Image processing, obstacle detecting, path planning, control, …

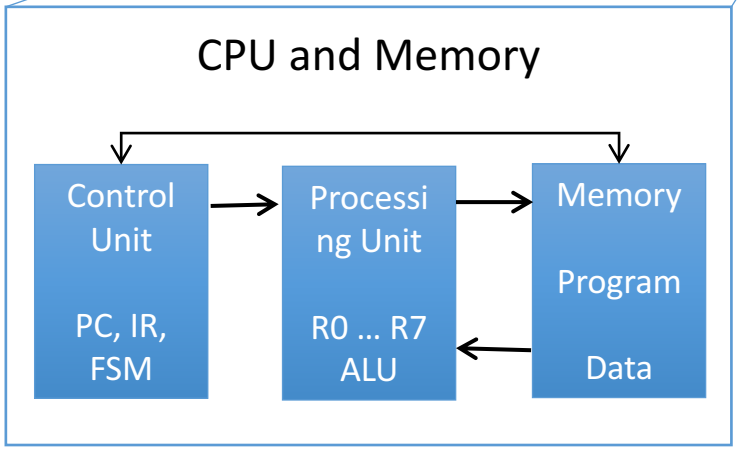
CPU and Memory



Complete system



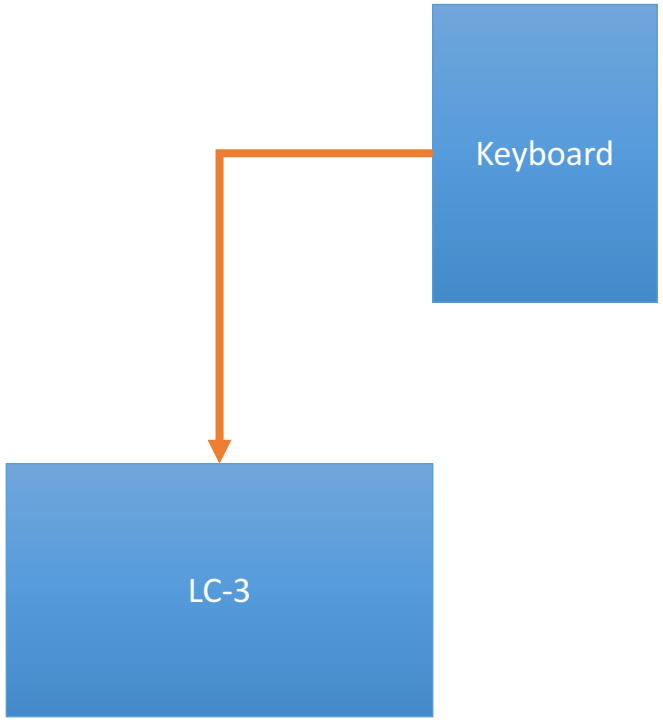
CPU & Memory
Controlling bipedal gait, Image processing, obstacle detecting, path planning, control, …



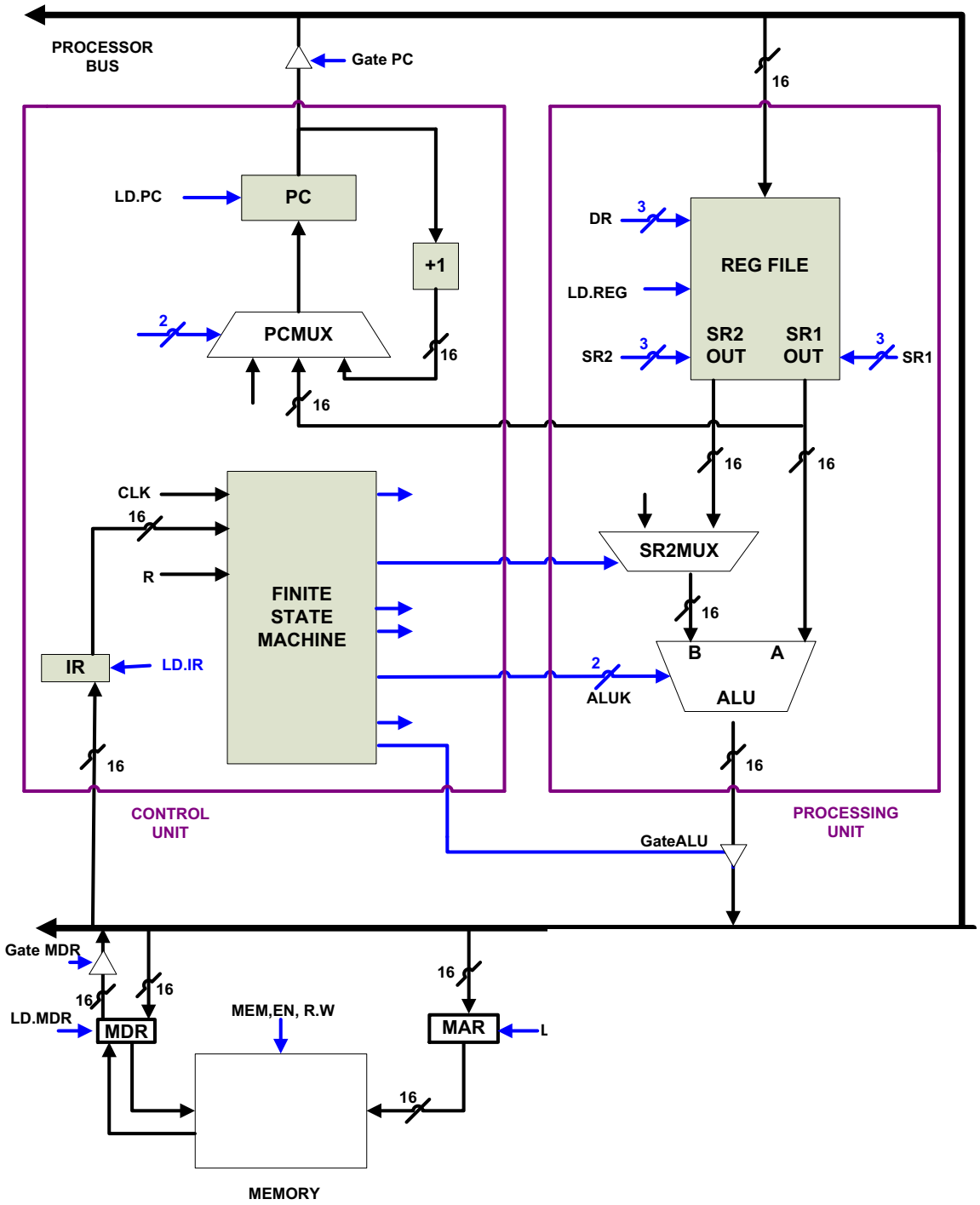
Many more examples

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

I/O Layout



- How to connect a keyboard to LC3?



how should we connect a keyboard to the computer?

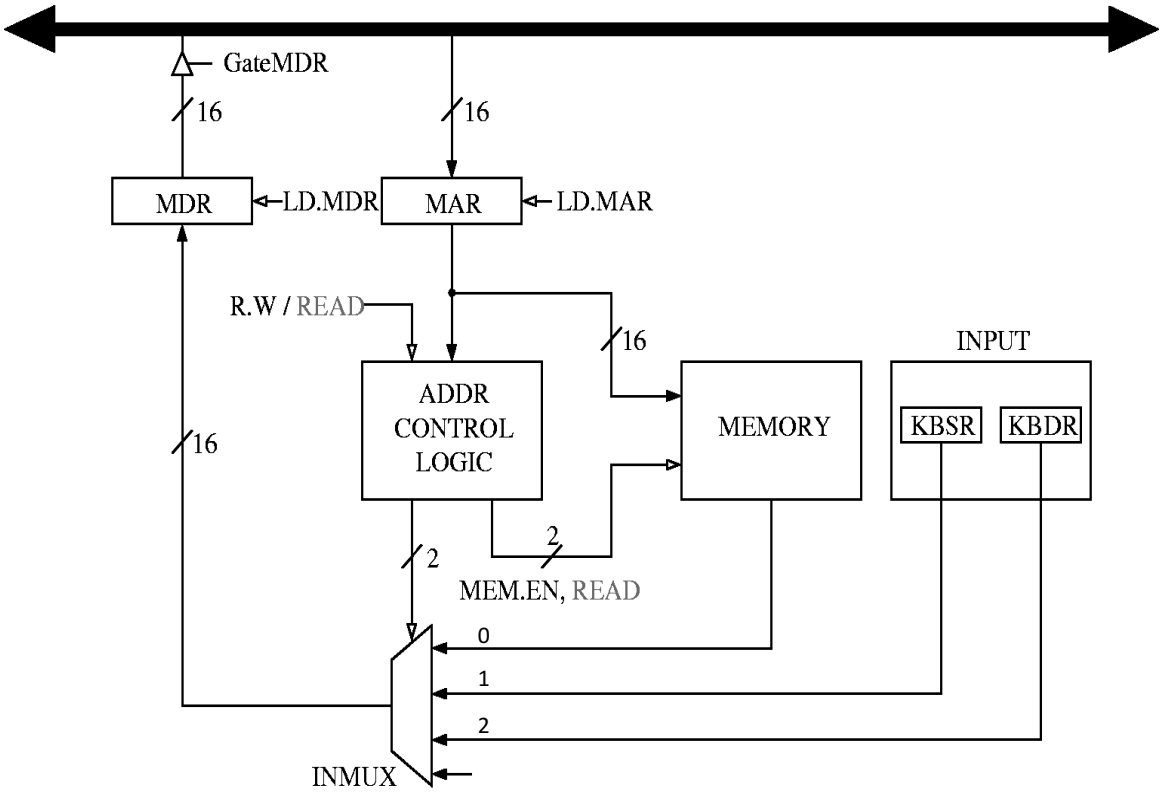
LC3 Memory: Memory mapped device registers

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

Circuit for memory mapped Inout



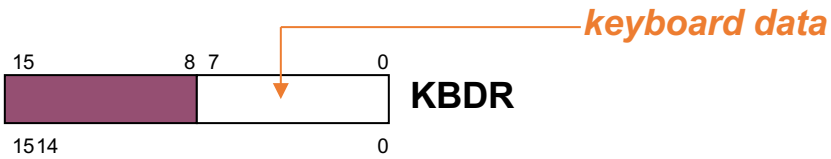
Conventional memory access: LD DR, addr

- MAR←addr
- MDR←MEM[MAR]
- DR←MDR

Memory-mapped input access: LD DR, xFE02

- MAR←xFE02
- MDR←KBDR
- DR←MDR

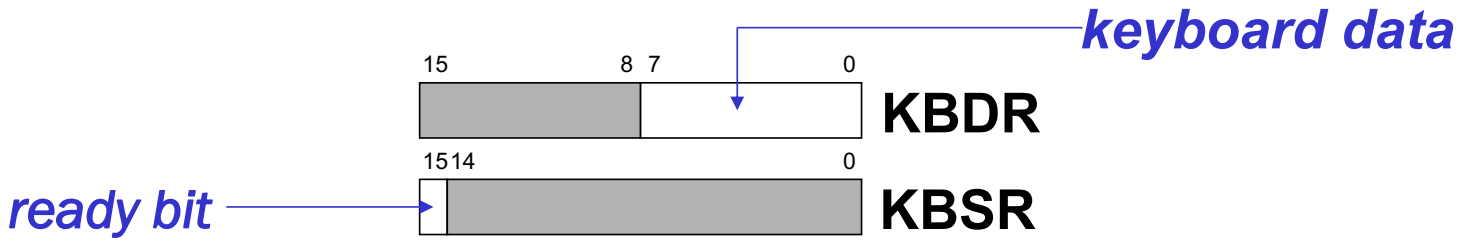
Reading Input (first attempt)




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

Does this work?

Handshaking using KBDR and KBSR

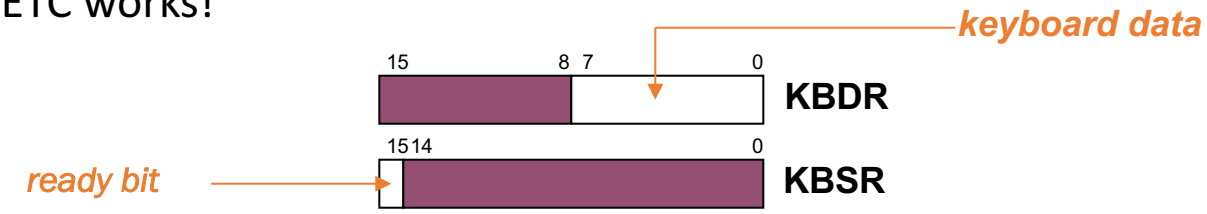


- When a char is typed by user in the keyboard
  - Its ASCII code is placed in KBDR[0:7]
  - KBSR[15] is set to 1 (ready bit)
  - Keyboard is disabled, i.e., any further keypress is ignored
- When KBDR is read by CPU
  - KBSR[15] is set to 0
  - Keyboard is enabled

This is part of the keyboard Hardware.

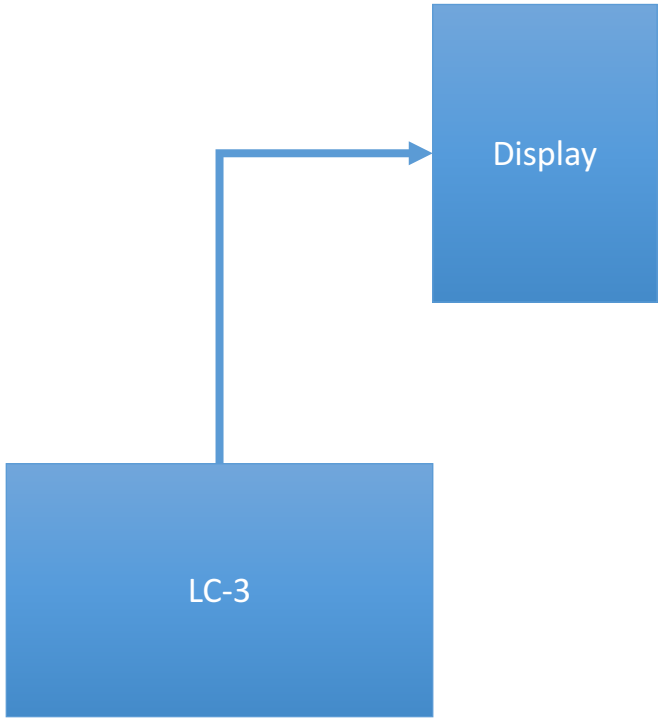
Reading Input the right way

This is how TRAP x20 = GETC works!



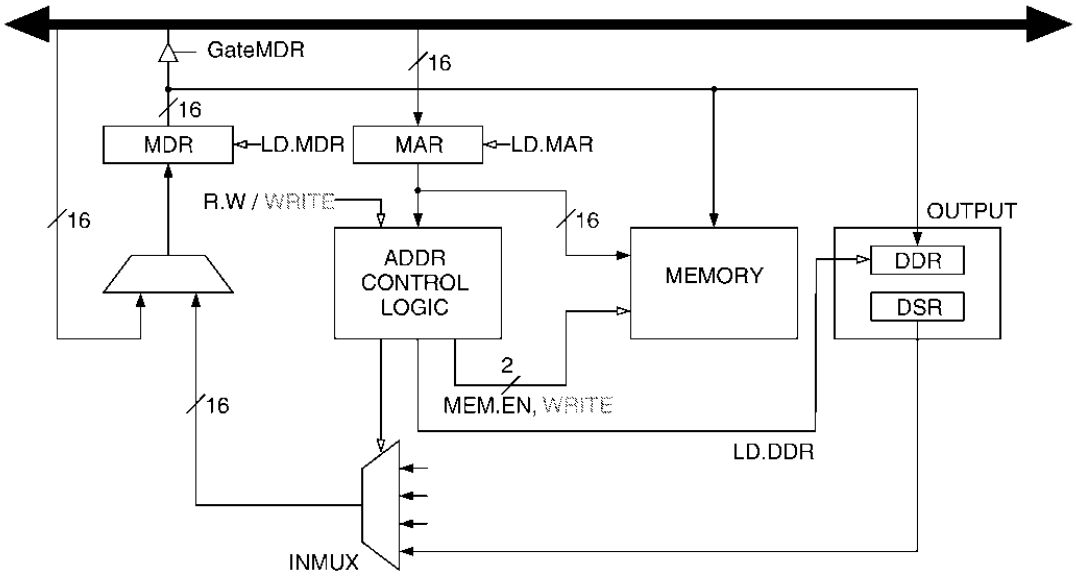

```
START           LDI     R1, KBSR_ADDR   ; Test for
                BRzp    START           ; character input
                LDI     R0, KBDR_ADDR
                BRnzp   NEXT_TASK       ; Go to the next
task
                ...
KBSR_ADDR       .FILL     xFE00         ; Address of KBSR
KBDR_ADDR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

I/O Layout



- How to connect a display to LC3?

Circuit for memory mapped output



Conventional write: ST SR, addr

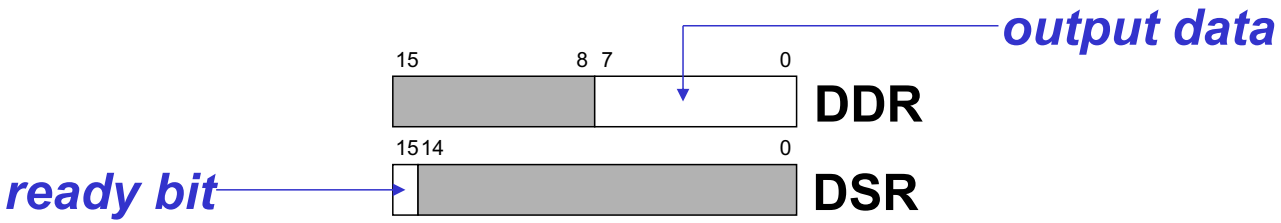
- MAR←addr
- MDR←SR
- Mem[MAR]←MDR

Memory-mapped input access: ST SR, xFE06

- MAR←xFE06
- MDR←SR
- DDR←MDR

Problem of asynchrony, again!

Handshaking using DDR and DSR

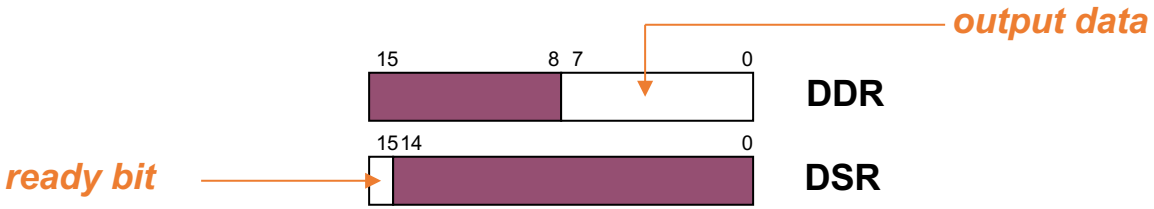


- When monitor is ready to display another char
  - DSR[15] is set to 1: (**ready bit**)
- When new char is written to DDR
  - DSR[15] is set to 0
  - Any other chars written to DDR are ignored
  - DDR[7:0] is displayed

This is part of the display hardware.

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 PUTS (display a stored string)
- Write code for ECHO (read a char and display it)
- Read Interrupt-driven I/O

Summary of concepts

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