ECE 220: Computer Systems & Programming

Lecture 2: I/O

Thomas Moon

January 18, 2024

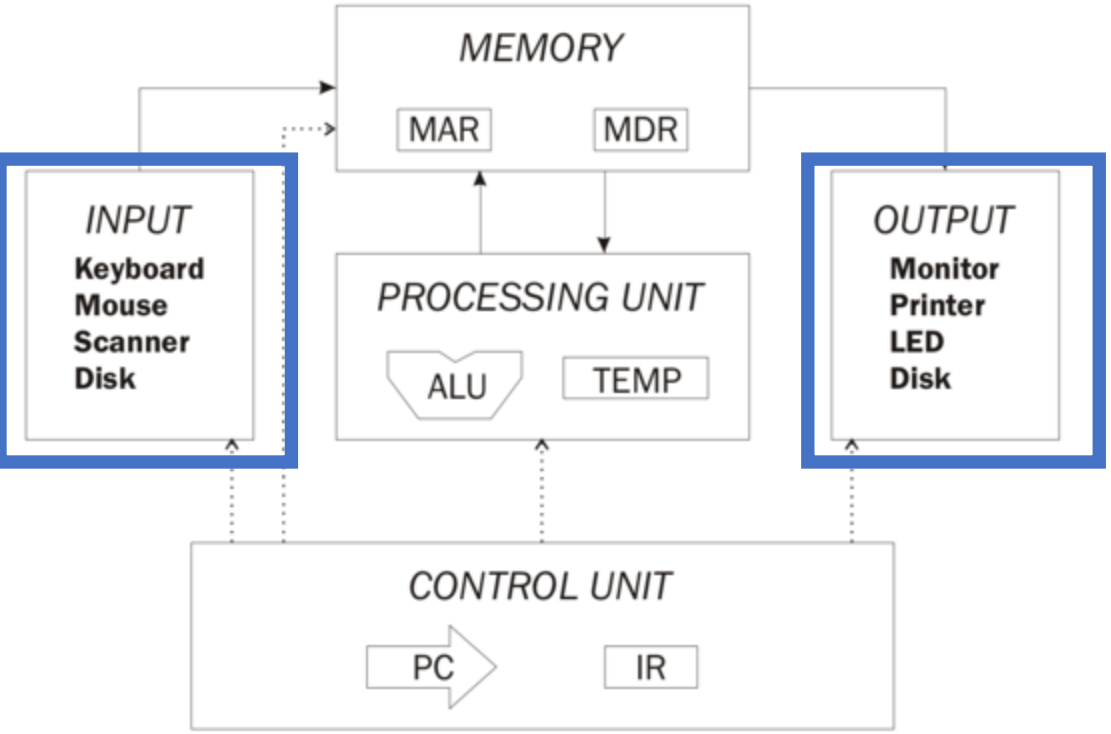
Announcements

- MP1 has been released.
- To retrieve MPs, you need to setup Git.

https://courses.grainger.illinois.edu/ece220/sp2024/pages/toolchain/git/setup/

- You can now reserve CBTF for the first mock quiz

ECE220…



Keyboard =

Display =

I/O Types

1. How *instruction* interacts with I/O device?

Memory-Mapped I/O

- Represent device registers as **memory** addresses
- Reuse memory instructions (LD/ST family)

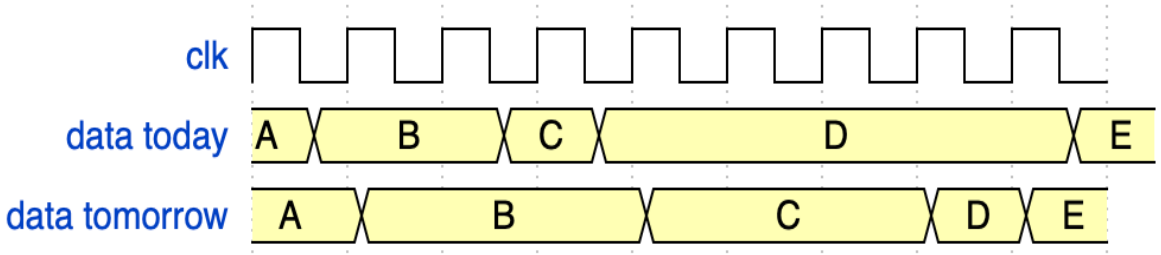
Special I/O Instructions

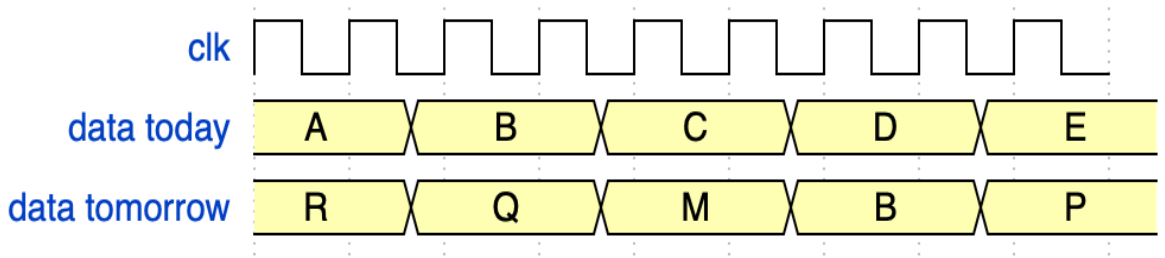
- **Extra** set of instructions for I/O (designated opcode)
- For example, your instruction set will look like
  ADD, NOT, AND, BR, …., ***KB, DP***

What if…
we want to add a mouse?

I/O Types

2. Transfer *timing*





Asynchronous

- Data rate less predictable (keyboard)
- Problem: missing data or multiple read/write
- Solution: Handshaking (ready bit)

Synchronous

- Data supplied at a fixed/predictable rate
- CPU reads/writes every X cycles

I/O Types

3. Who *controls* the interaction?

Polling

- Processor controls the interaction
- Keep asking whether the I/O data is ready
- *"Are you ready? Are you ready? …"*

Interrupt-Driven

- I/O controls the interaction
- Processor is interrupted by announcement from I/O
- *"Wake me up when you are ready."*

I/O Types of LC-3

Memory-Mapped I/O vs Special I/O Instructions

→KBSR, KBDR, DSR, DDR

Asynchronous vs Synchronous

→ Handshake thru status registers

The last lecture!

Polling vs Interrupt-Driven

LC-3 I/O Device Registers

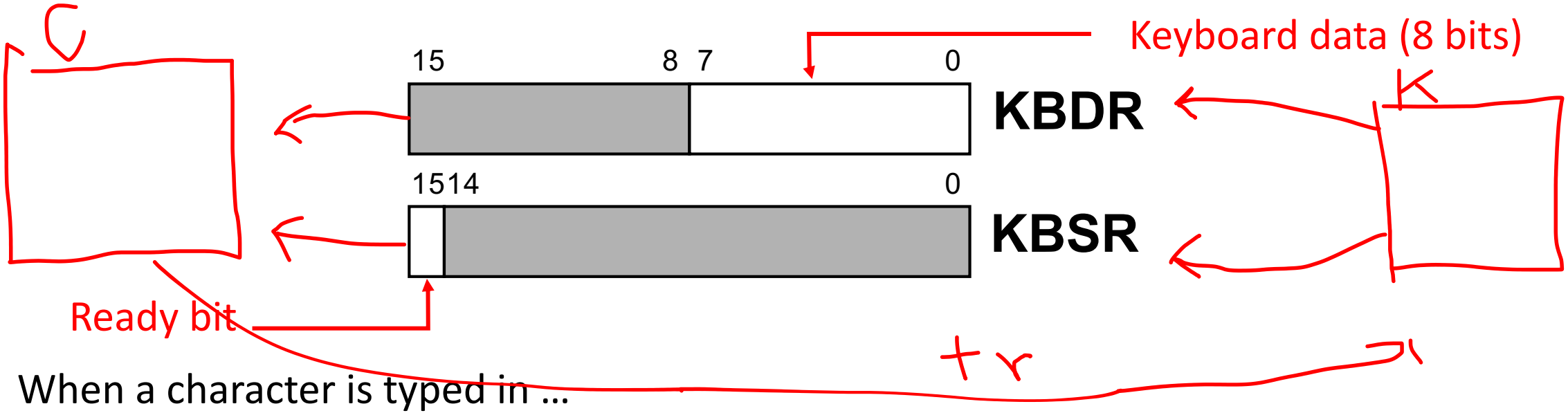
Keyboard

- **KBDR** : store ASCII value entered from *keyboard*
- **KBSR** : let processor know a new value is entered

Monitor

- **DDR** : store ASCII value to be displayed on *monitor*
- **DSR** : let processor know a new value is ready to be displayed

Input from keyboard (Handshaking)



When a character is typed in …

1. Its ASCII code is placed in bits [7:0] of KBDR (bits [15:8] are always zero)
2. The “ready bit” (KBSR[15]) is set to one (by keyboard electronic circuits)
3. Keyboard is disabled -- any typed characters will be ignored

When KBDR is read …

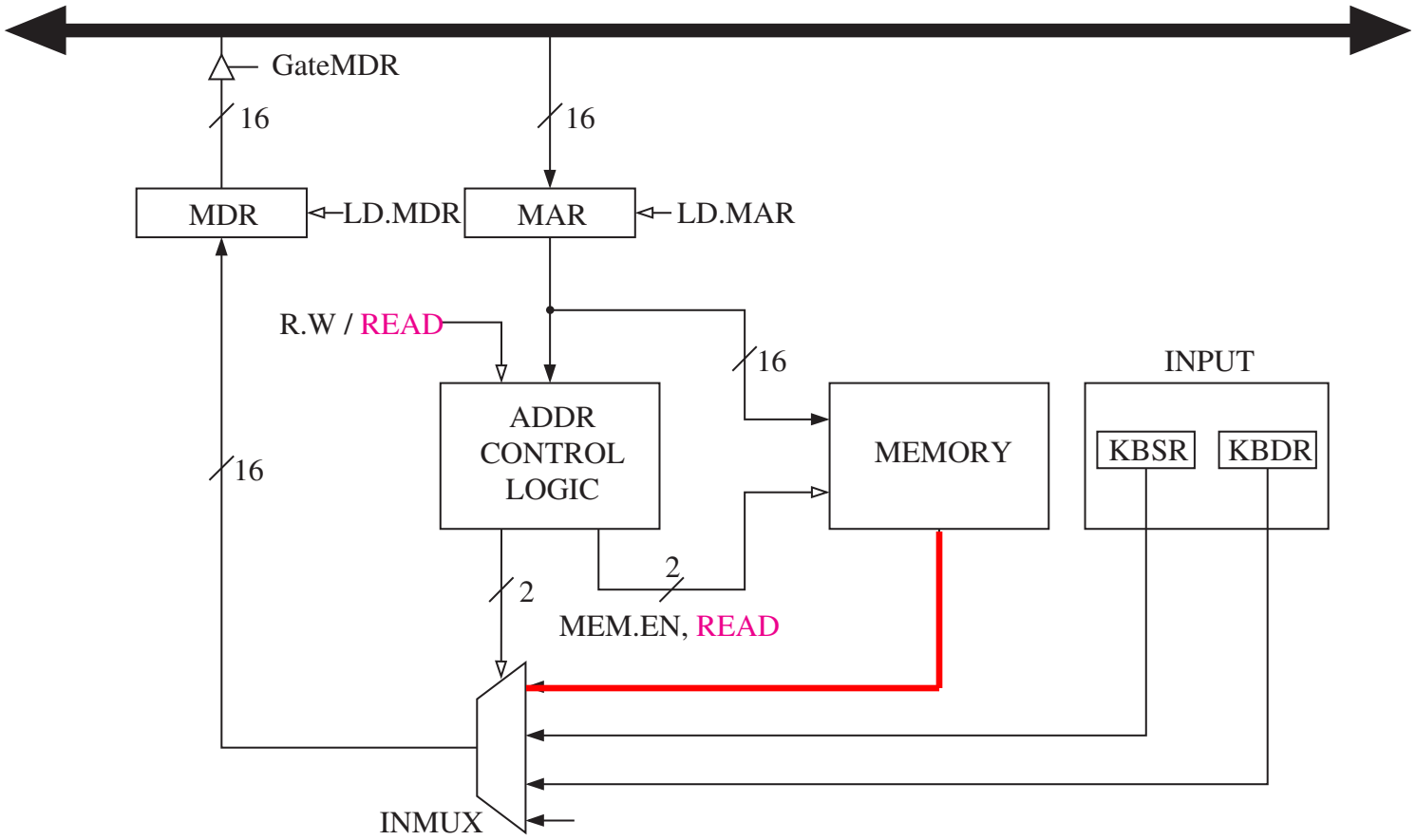
1. KBSR[15] is set to zero (by keyboard electronic circuits)
2. Keyboard is enabled

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		

- The device registers (KBDR, KBSR, DDR, DSR) are mapped to the memory address.
- The device registers are physically separated from the memory.

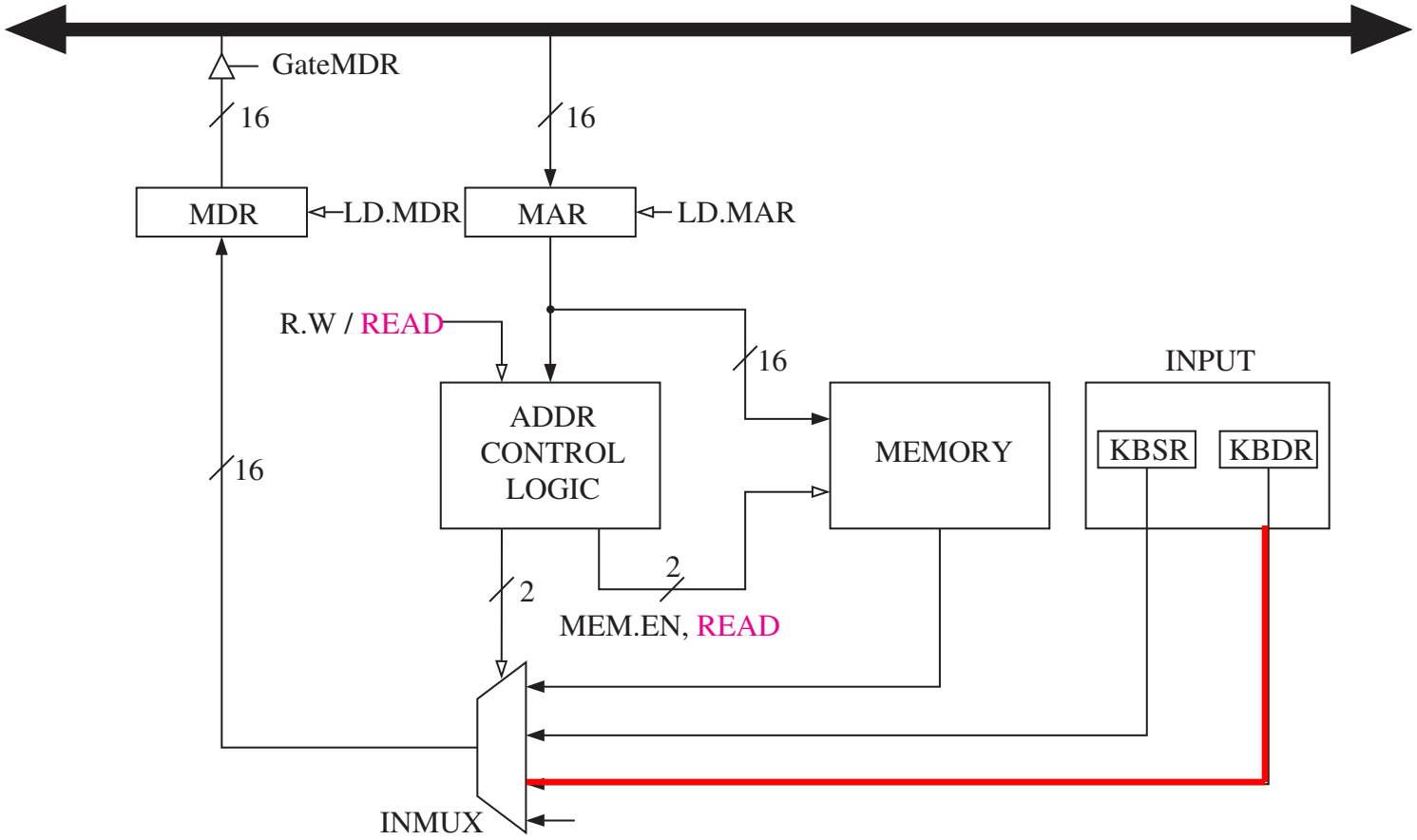
Circuit for Memory-mapped I/O



Read data from Memory at address X

1. MAR ← X
2. READ signal
3. MDR ← MEM[MAR]

Circuit for Memory-mapped I/O



Read data at KBDR

1. MAR ← xFE02
2. READ signal
3. MDR ← KBDR

*Address control logic decodes the address and select either memory or KBSR/KBDR.

Q. When KBDR is ready, the value of KBSR is

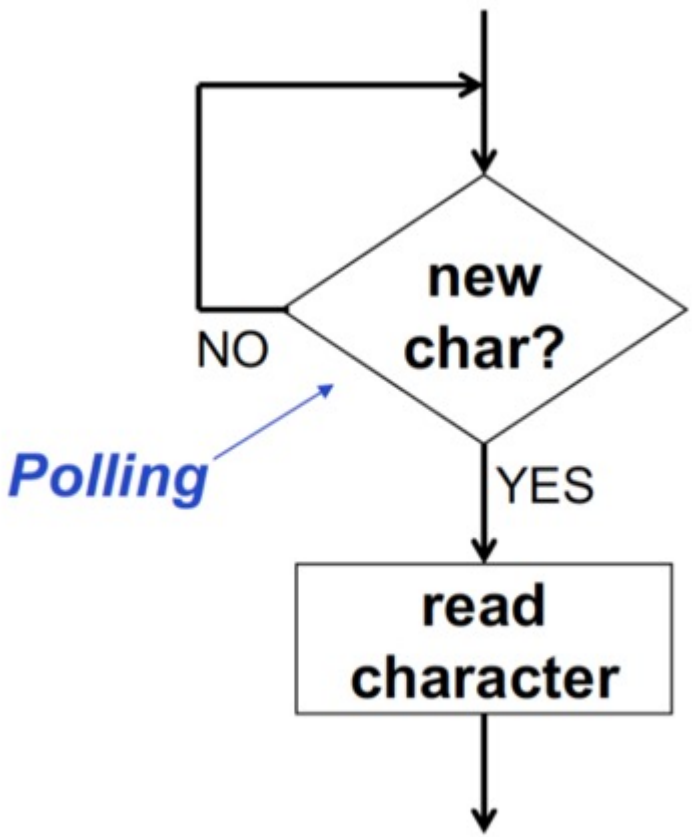
1. Negative
2. Zero
3. Positive
4. Unknown

Q. Which instruction updates the condition codes (NZP)?

1.	ADD	R1,R1,#1	R1 <- R1 + #1
2.	STR	R4,R1,#1	mem[R1+1] <- R4
3.	LDI	R1,LABEL	R1 <- mem[mem[LABEL]]
4.	NOT	R3,R4	R3 <- NOT(R4)

hint: each time R0-R7 is written, NZP is updated

Basic Input Routine (by polling)



- Read a single character from keyboard.

  1. Load KBSR to a register (R0-R7)
  2. Check its MSB by sign

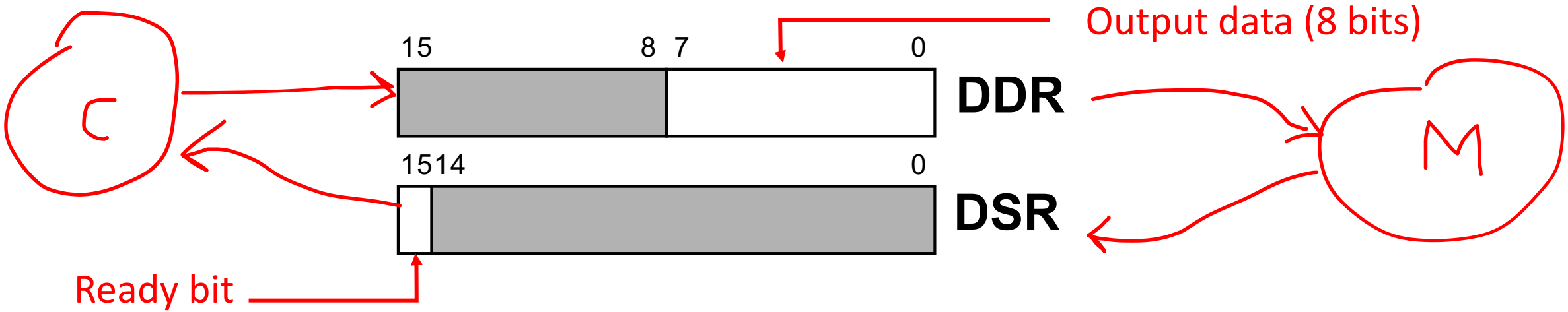
     → *Z or P: repeat 1*

     → *N: Load KBDR to a register*

input.asm

1. How do you want to read KBSR/KBDR?
   (i.e. LD or LDI or LDR or LEA)
2. How do you want to branch?
   (which BR?)

Output to Monitor (Handshaking)



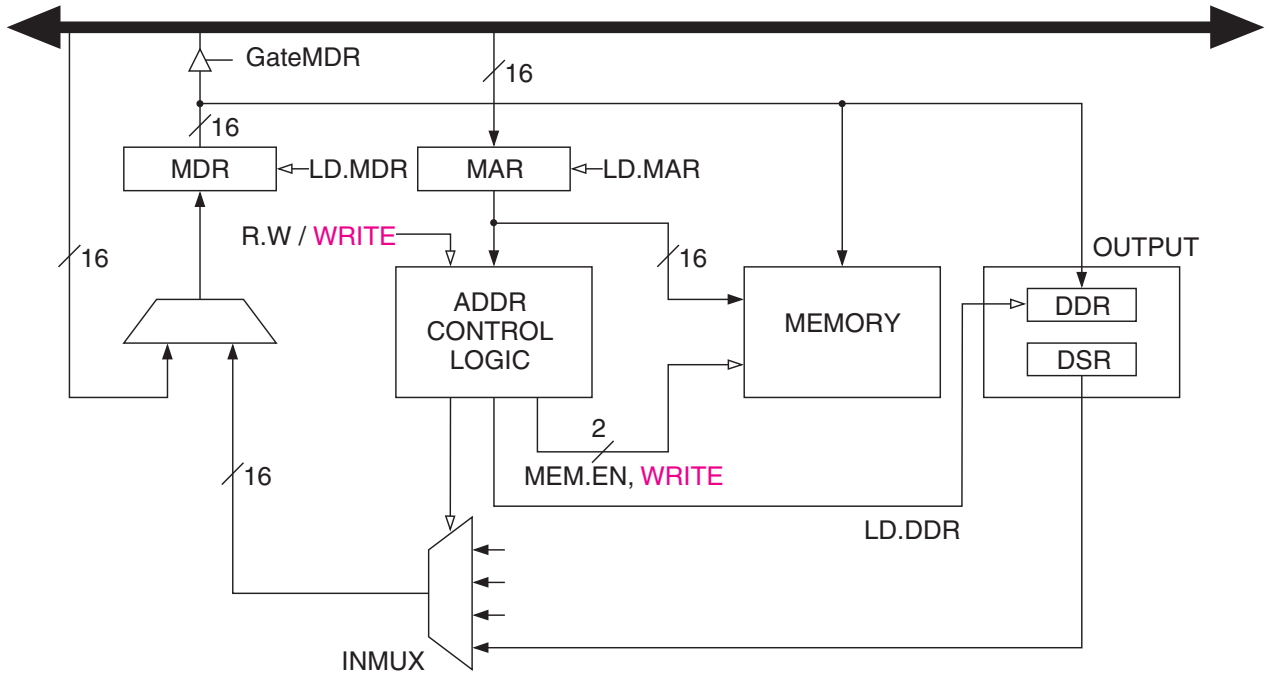
When Monitor is ready to display another character…

1. The “ready bit” (DSR[15]) is set to one

When data is written to DDR

1. DSR[15] is set to zero
2. Character in DDR[7:0] is displayed
3. Any other character written to DDR is ignored, while DSR[15] is zero

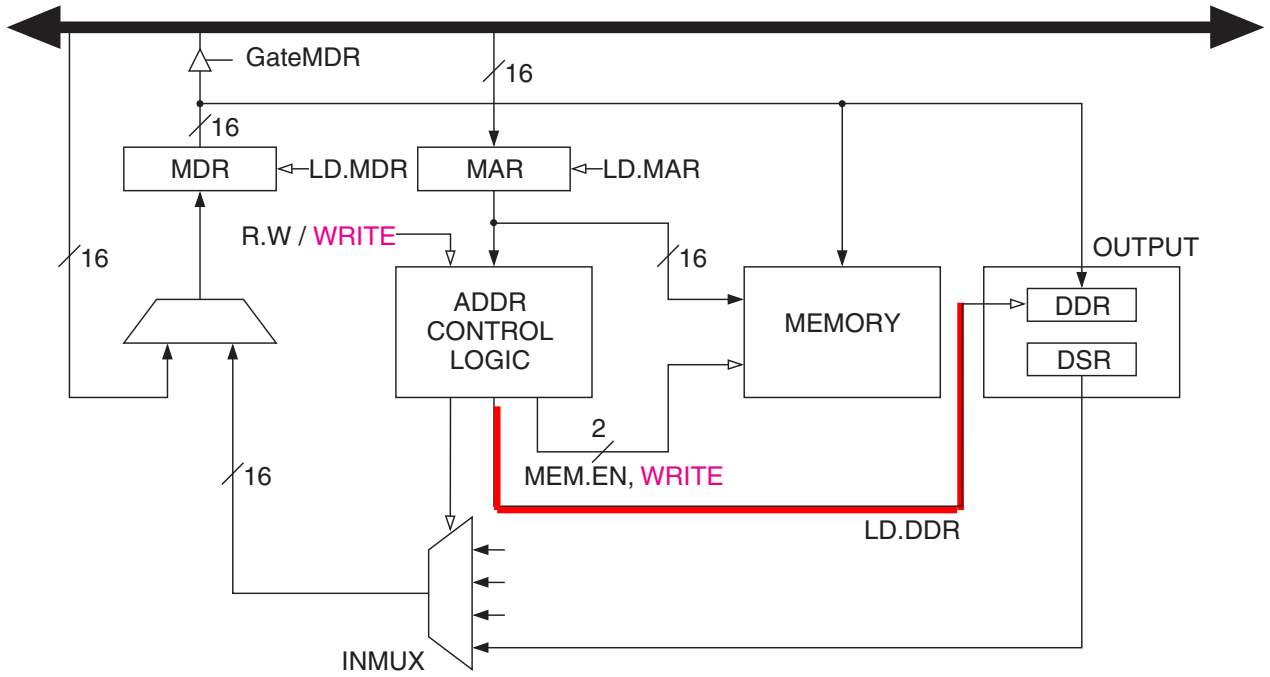
Circuit for Memory-mapped I/O



Write data to Memory at address Y

1. MAR ← Y, MDR ← Data
2. WRITE signal
3. MEM[MAR] ← Data

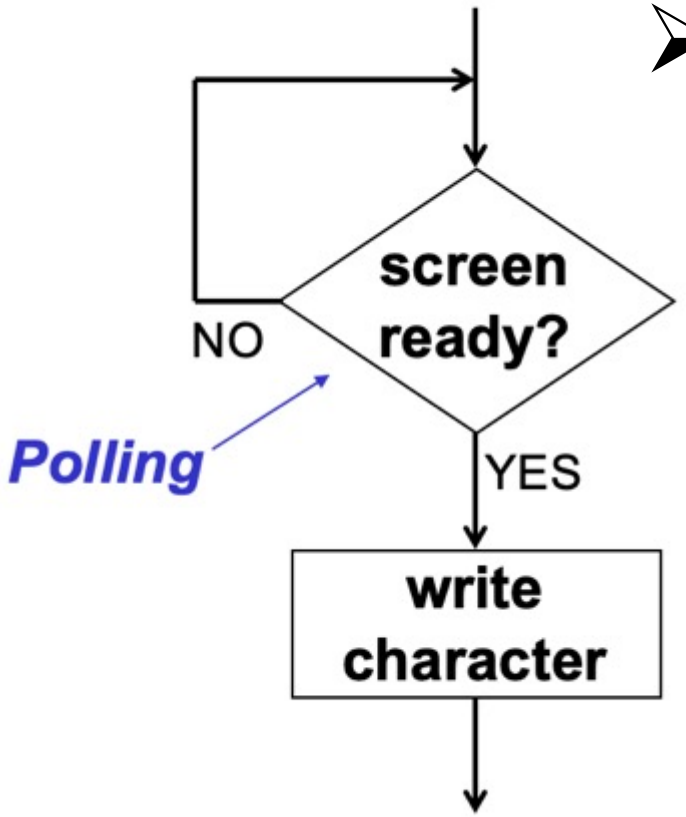
Circuit for Memory-mapped I/O



Write data to DDR

1. MAR ← xFE06, MDR ← Data
2. LD.DDR signal
3. DDR ← Data

Basic Output Routine



- Display a single character to monitor.

1. Load DSR to a register (R0-R7)
2. Check its MSB sign
   → *Z or P: repeat 1*
   → *N: Store a character to DDR*

output.asm

1. How do you want to read DSR?
2. How do you want to write DDR?
3. How do you want to branch?
   (which BR?)

Echo keyboard input (input.asm + output.asm)

LC-3 TRAP Routines for Handling I/O

- TRAP routines for input

Vector	Symbol	Routine
x20	GETC	Read a single character (no echo)
x23	IN	Print prompt to console, read and echo character from keyboard

- TRAP routines for output

Vector	Symbol	Routine
x21	OUT	Output a character to the monitor
x22	PUTS	Write a string to the console

GETC/OUT vs input.asm & output.asm

```
          OS_R2 x0449 x0000 NOP
          OS_R3 x044A x0000 NOP
          OS_R7 x044B x0490 BRZ    x04DC
       TRAP_GETC x044C xA1F1 LDI   R0,OS_KBSR
                 x044D x07FE BRZP  TRAP_GETC
                 x044E xA1F0 LDI   R0,OS_KBDR
                 x044F xC1C0 RET
        TRAP_OUT x0450 x33F4 ST    R1,TOUT_R1
   TRAP_OUT_WAIT x0451 xA3EE LDI   R1,OS_DSR
                 x0452 x07FE BRZP  TRAP_OUT_WAIT
                 x0453 xB1ED STI   R0,OS_DDR
                 x0454 x23F0 LD    R1,TOUT_R1
                 x0455 xC1C0 RET
```

Very similar except a few things…
More on Lecture 3!