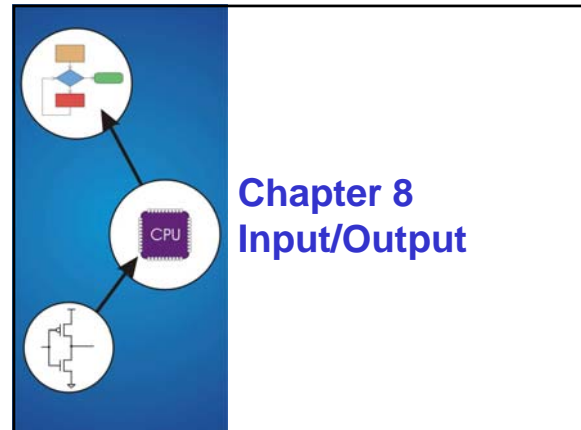




Introduction to Computer Engineering

ECE/CS 252, Fall 2010
 Prof. Mikko Lipasti
 Department of Electrical and Computer Engineering
 University of Wisconsin – Madison



Chapter 8 Input/Output

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

I/O: Connecting to Outside World

So far, we've learned how to:

- compute with values in registers
- load data from memory to registers
- store data from registers to memory

But where does data in memory come from?

And how does data get out of the system so that humans can use it?

8-3

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

I/O: Connecting to the Outside World

Types of I/O devices characterized by:

- **behavior:** input, output, storage
 - input: keyboard, motion detector, network interface
 - output: monitor, printer, network interface
 - storage: disk, CD-ROM
- **data rate:** how fast can data be transferred?
 - keyboard: 100 bytes/sec
 - disk: 30 MB/s
 - network: 1 Mb/s - 1 Gb/s

8-4

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

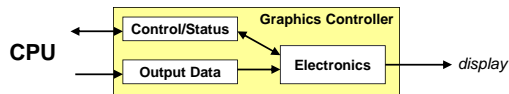
I/O Controller

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, bits to/from disk, characters from keyboard

8-5

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Programming Interface

How are device registers identified?

- Memory-mapped vs. special instructions

How is timing of transfer managed?

- Asynchronous vs. synchronous

Who controls transfer?

- CPU (polling) vs. device (interrupts)

8-6

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Memory-Mapped vs. I/O Instructions

Instructions

- designate opcode(s) for I/O
- register and operation encoded in instruction

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IO										Device				Op	

Memory-mapped

- assign a memory address to each device register
- use data movement instructions (LD/ST) for control and data transfer

8-7

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Transfer Timing

I/O events generally happen much slower than CPU cycles.

Synchronous

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

Asynchronous

- data rate less predictable
- CPU must synchronize with device, so that it doesn't miss data or write too quickly

8-8

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Transfer Control

Who determines when the next data transfer occurs?

Polling

- CPU keeps checking status register until new data arrives OR device ready for next data
- "Are we there yet? Are we there yet? Are we there yet?"

Interrupts

- Device sends a special signal to CPU when new data arrives OR device ready for next data
- CPU can be performing other tasks instead of polling device.
- "Wake me when we get there."

8-9

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Recap: Programming Interface

How are device registers identified?

- Memory-mapped vs. special instructions

How is timing of transfer managed?

- Asynchronous vs. synchronous

Who controls transfer?

- CPU (polling) vs. device (interrupts)

8-10

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

LC-3

Memory-mapped I/O (Table A.3)

Location	I/O Register	Function
xFE00	Keyboard Status Reg (KBSR)	Bit [15] is one when keyboard has received a new character.
xFE02	Keyboard Data Reg (KBDR)	Bits [7:0] contain the last character typed on keyboard.
xFE04	Display Status Register (DSR)	Bit [15] is one when device ready to display another char on screen.
xFE06	Display Data Register (DDR)	Character written to bits [7:0] will be displayed on screen.

Asynchronous devices

- synchronized through status registers

Polling and Interrupts

- the details of interrupts will be discussed in Chapter 10

8-11

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Input from Keyboard

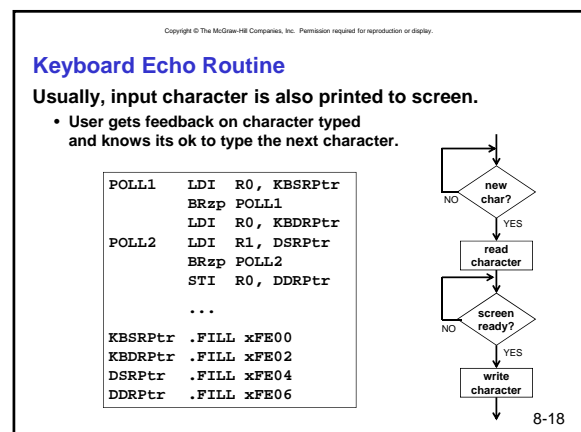
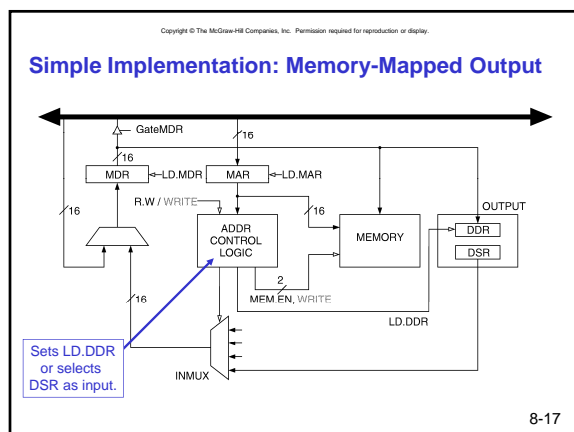
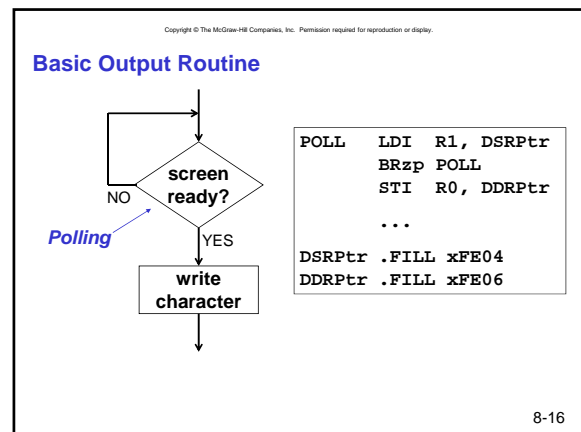
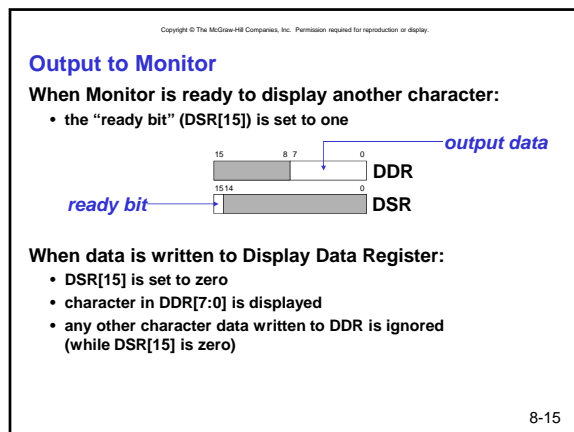
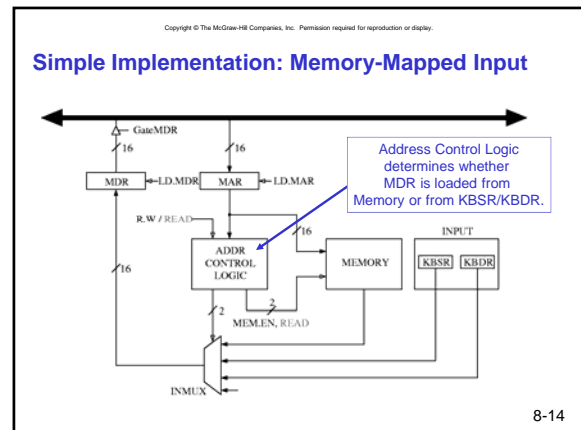
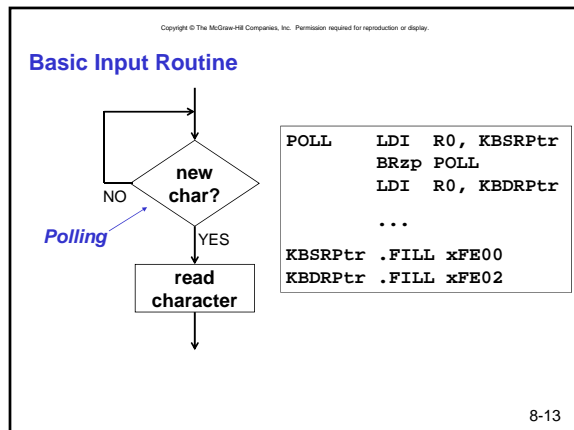
When a character is typed:

- its ASCII code is placed in bits [7:0] of KBDR (bits [15:8] are always zero)
- the "ready bit" (KBSR[15]) is set to one
- keyboard is disabled -- any typed characters will be ignored

When KBDR is read:

- KBSR[15] is set to zero
- keyboard is enabled

8-12



Interrupt-Driven I/O

External device can:

- (1) Force currently executing program to stop;
- (2) Have the processor satisfy the device's needs; and
- (3) Resume the stopped program as if nothing happened.

Why?

- Polling consumes a lot of cycles, especially for rare events – these cycles can be used for more computation.
- Example: Process previous input while collecting current input. (See Example 8.1 in text.)

8-19

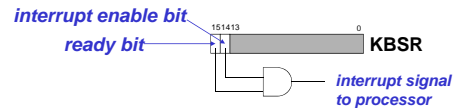
Interrupt-Driven I/O

To implement an interrupt mechanism, we need:

- A way for the I/O device to **signal** the CPU that an interesting event has occurred.
- A way for the CPU to **test** whether the **interrupt signal is set** and whether its **priority is higher** than the current program.

Generating Signal

- Software sets "interrupt enable" bit in device register.
- When ready bit is set and IE bit is set, interrupt is signaled.



8-20

Priority

Every instruction executes at a stated level of urgency.

LC-3: 8 priority levels (PL0-PL7)

- Example:
 - Payroll program runs at PL0.
 - Nuclear power plant control program runs at PL6.
- It's OK for PL6 device to interrupt PL0 program, but not the other way around.

Priority encoder selects highest-priority device, compares to current processor priority level, and generates interrupt signal if appropriate.

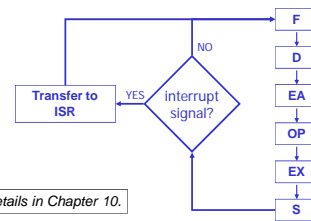
8-21

Testing for Interrupt Signal

CPU looks at signal between STORE and FETCH phases.

If not set, continues with next instruction.

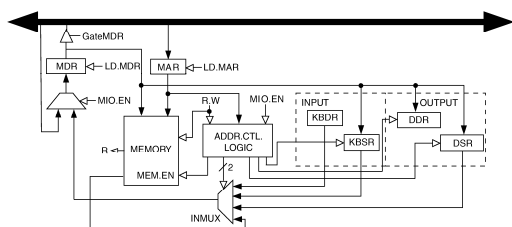
If set, transfers control to interrupt service routine.



More details in Chapter 10.

8-22

Full Implementation of LC-3 Memory-Mapped I/O



Because of interrupt enable bits, status registers (KBSR/DSR) must be written, as well as read.

8-23

Review Questions

- What is the danger of not testing the DSR before writing data to the screen?
 - Skip or garble display of previous character
- What is the danger of not testing the KBSR before reading data from the keyboard?
 - Could read previous character, or even garbage data

8-24

Review Questions

- Do you think polling is a good approach for other devices, such as a disk or a network interface?
 - In general, no: inefficient, complex for multiple devices, blocking
- What is the advantage of using LD/STI for accessing device registers?
 - Device addresses far away from code: 9-bit PC offset insufficient
 - LD+LDR requires two instructions for each access
 - Or requires an additional register (LD once, then LDR for each polling attempt)

8-25

Next lecture: System Calls

Certain operations require **specialized knowledge** and **protection**:

- specific knowledge of I/O device registers and the sequence of operations needed to use them
- I/O resources shared among multiple users/programs; a mistake could affect lots of other users!

Not every programmer knows (or wants to know) this level of detail

Provide **service routines** or **system calls** (part of operating system) to safely and conveniently perform low-level, privileged operations

8-26

Summary

Behavior and data rate of I/O device

Asynchronous vs. synchronous

Polled vs. interrupt-driven

Programmed vs. memory-mapped

Control registers, data registers

8-27