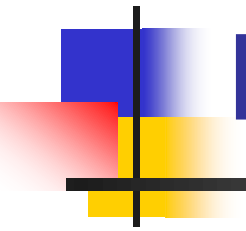
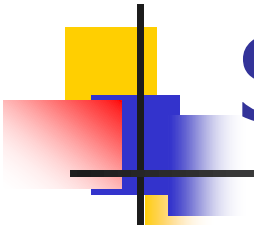


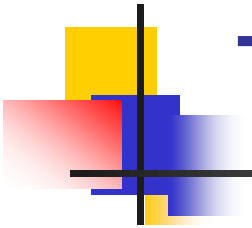
Microprocessor Systems

Dr. Gökhan İnce



Syllabus

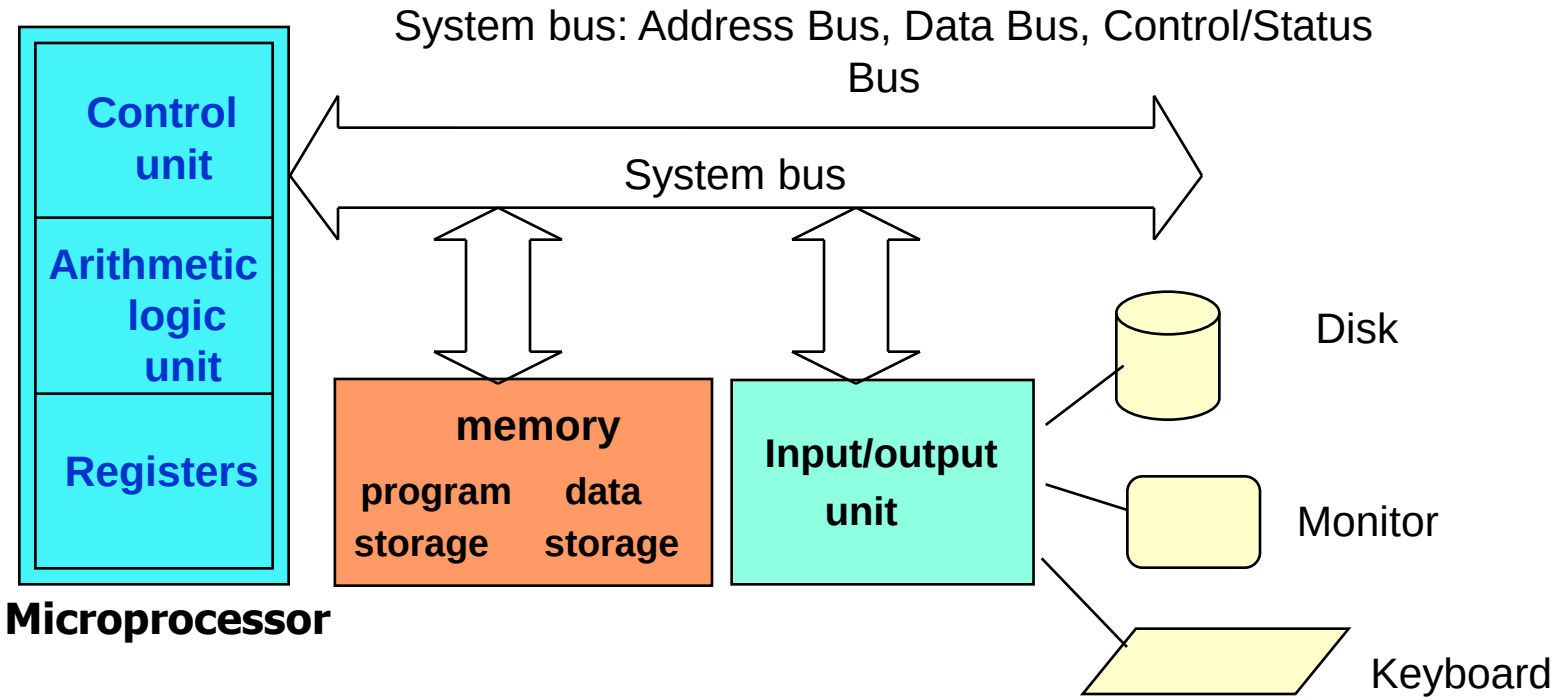
1. Introduction, Number Systems
2. Computer Overview - Memory
3. Memory Design
4. Quiz 1, CPU overview, Instruction format
5. Addressing methods
6. Instruction types
7. Instruction types - cntd
8. Midterm Exam 1
9. Parallel communication interface
10. Serial communication interface
11. Quiz 2, Subroutines, Interrupts, Stack, Coding techniques
12. Coding examples and applications
13. Midterm Exam 2
14. Development of Microprocessor Based Designs



Topics

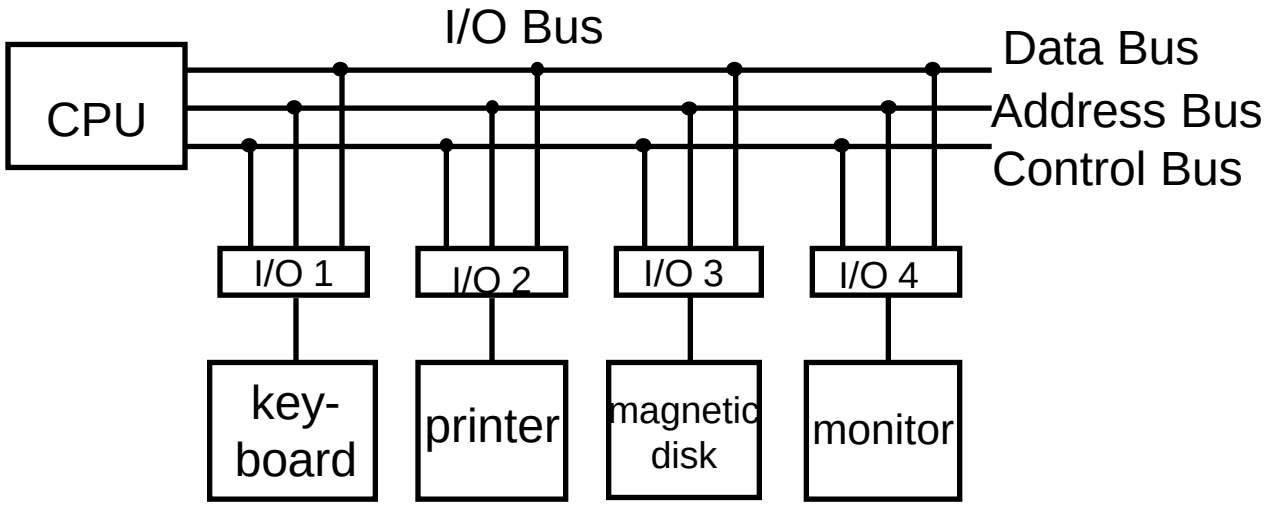
- I/O Interface
- I/O Transfer Synchronization
- Parallel Communication

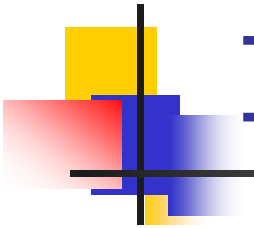
Computer Organization



I/O Interface

- Peripheral devices (also called I/O devices) are pieces of equipment that exchange data with a CPU
  - Examples of I/O devices include switches, push-buttons, light-emitting diodes (led), monitors, printers, modems, keyboards, and disk drives.
- Interface chips are needed to resolve the differences between CPU and I/O devices
  - Provides a method for transferring information between internal storage (such as memory and CPU registers) and external I/O devices





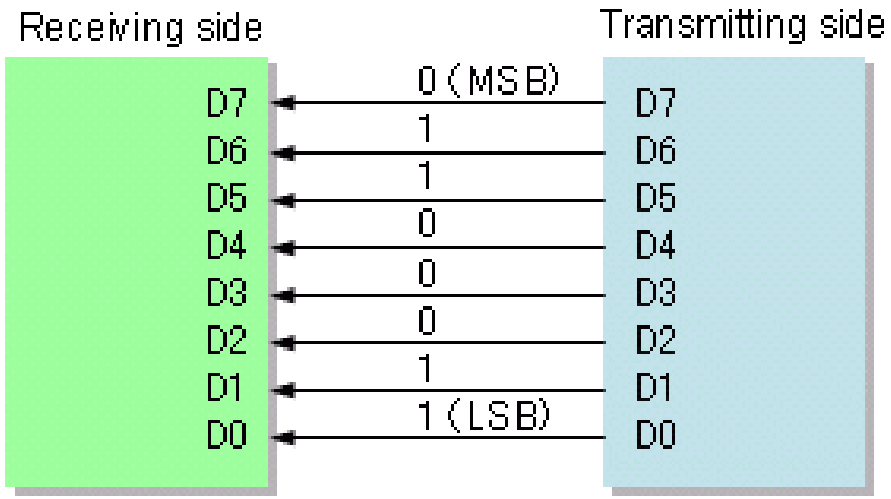
I/O Interfacing

- Resolves the differences between the computer and peripheral devices
  - Peripherals are electromechanical and electromagnetic devices, CPU and memory are electronic devices
  - Data codes and formats in peripherals are different
- A synchronization mechanism is needed.
  - Data transfer rate of peripherals are usually slower
- Data transfer between an I/O device and the CPU can be proceeded bit-by-bit (serial) or in multiple bits (parallel).

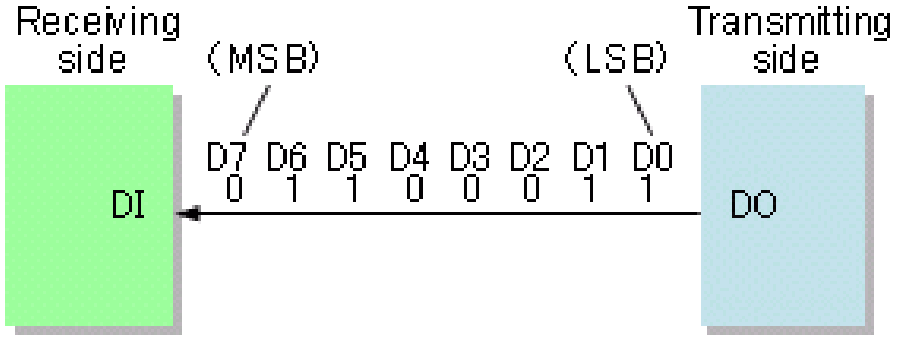
I/O Interfacing

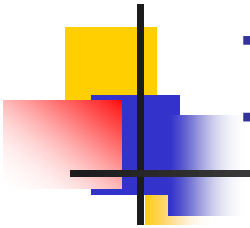
- The I/O communication can be Parallel or Serial

Parallel interface example



Serial interface example (MSB first)



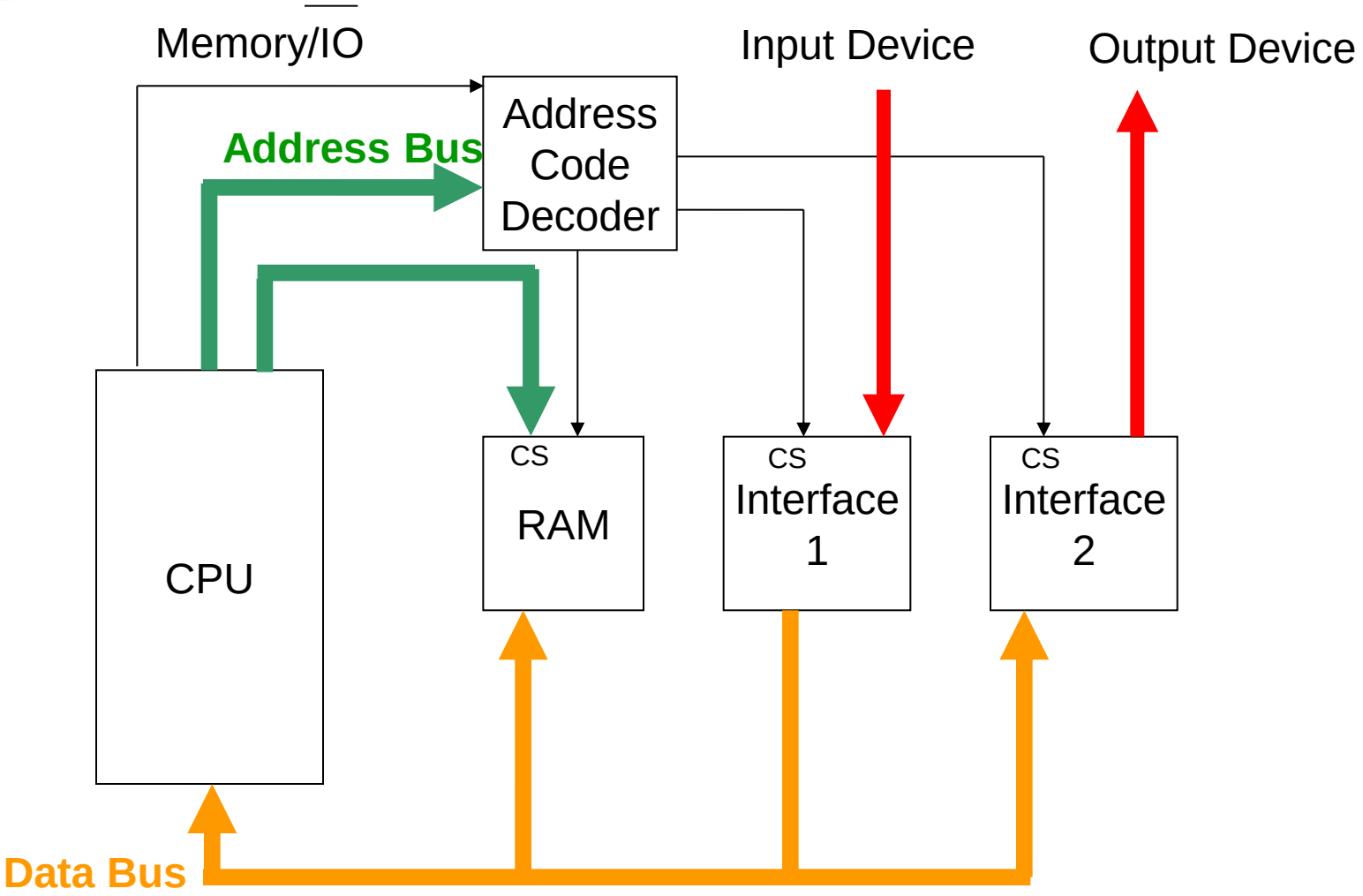


I/O Addresses

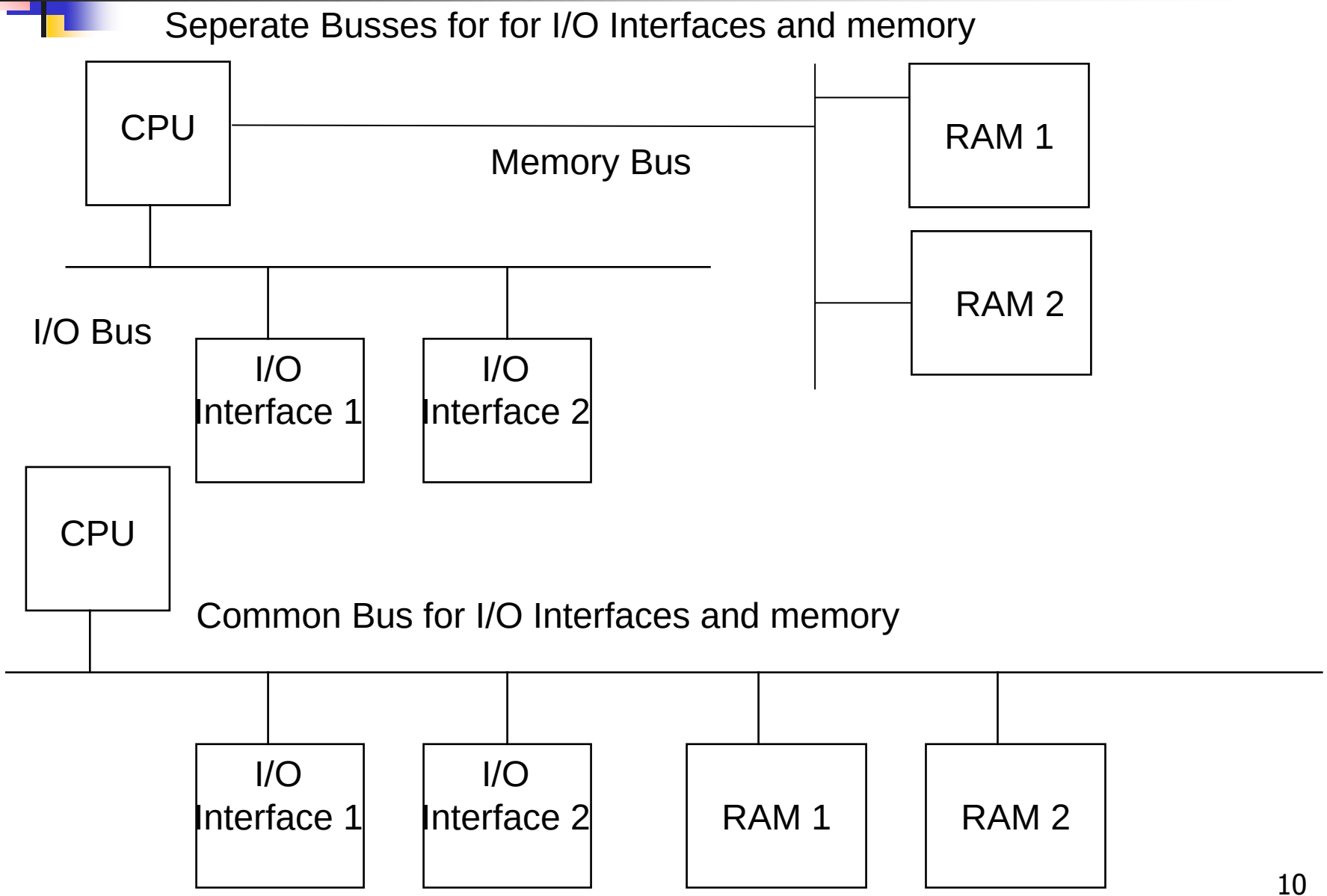
- Isolated I/O Map
  - Separate I/O read/write control lines in addition to memory read/write control lines
  - Distinct input and output instructions
  - Separate (isolated) memory and I/O address spaces

- Memory Mapped I/O
  - A single set of read/write control lines (no distinction between memory and I/O transfer)
  - No specific input or output instruction
    - The same memory reference instructions can be used for I/O transfers
  - Memory and I/O addresses share the common address space
    - reduces memory address range available
  - Considerable flexibility in handling I/O operations

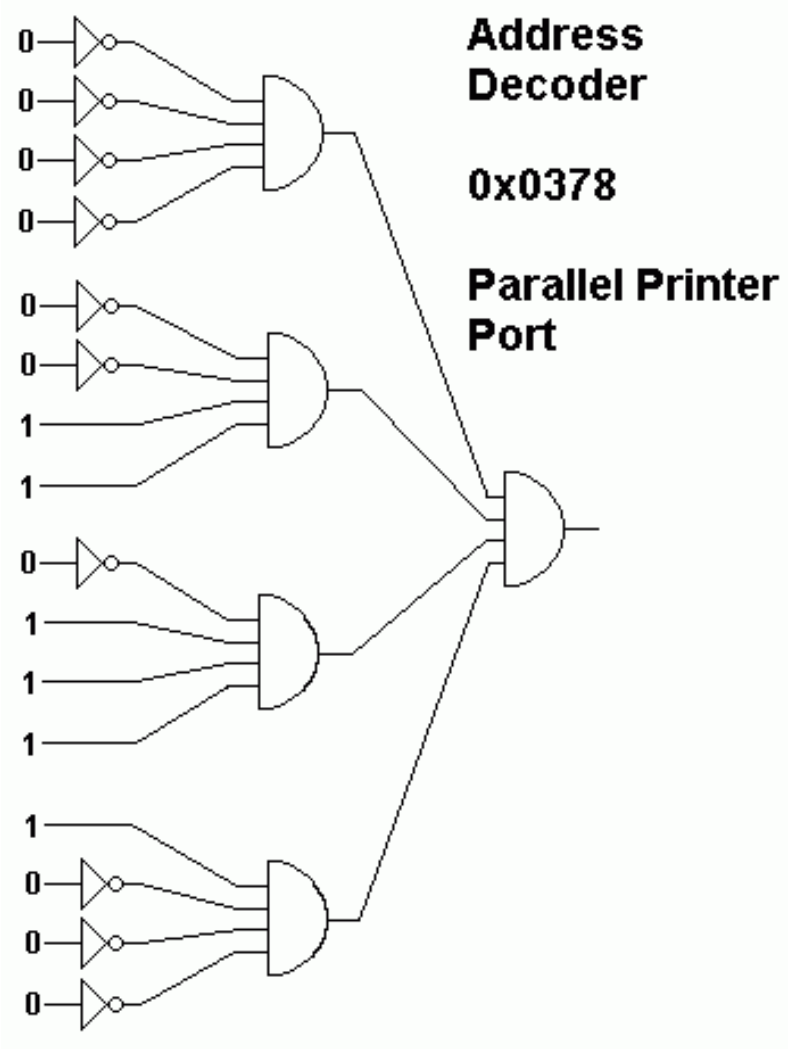
Isolated I/O Map-based Interfacing

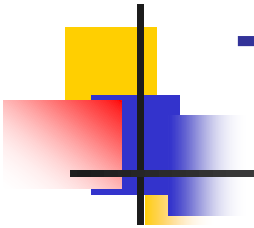


Connection of I/O Interfaces



Address Decoder for Memory-mapped I/O interface



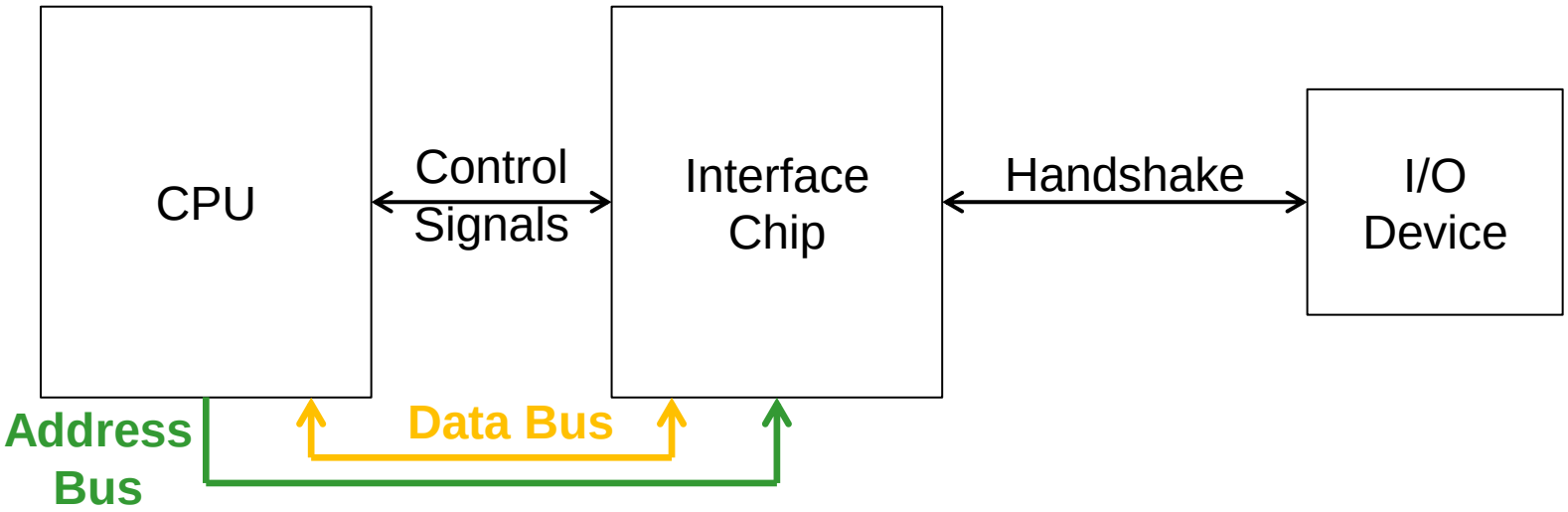


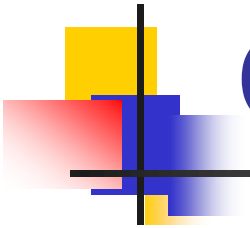
Topics

- I/O Interface
- I/O Transfer Synchronization
- Parallel Communication

I/O Transfer Synchronization

- The role of the interface chip
  - Synchronizing data transfer between CPU and interface chip.
    - Programmed I/O
    - Interrupt driven I/O
    - Direct Memory Access (DMA)
  - Synchronizing data transfer between interface chip and I/O device.
    - Handshaking method





CPU and I/O Interfaces

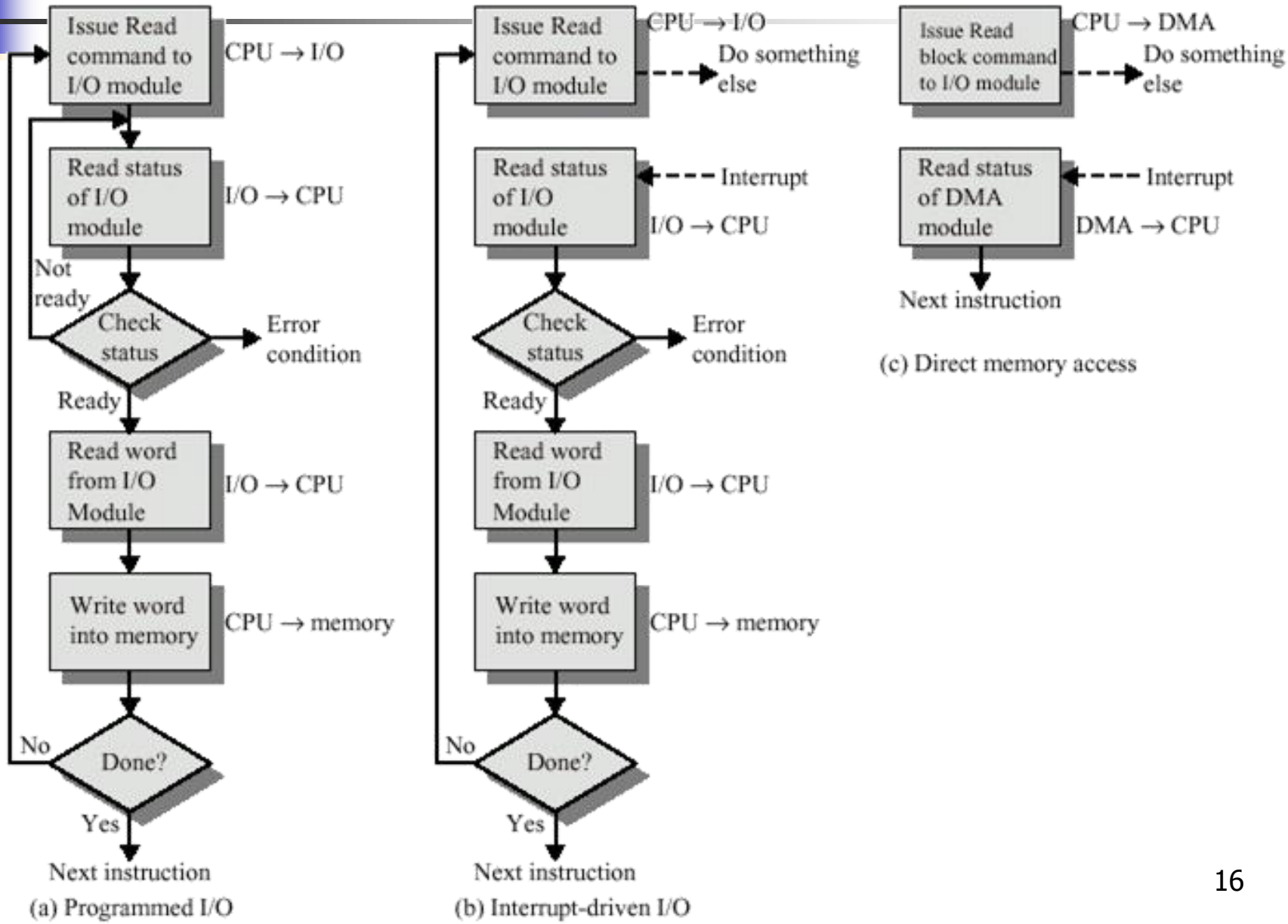
- Programmed I/O
  - Sensing status
  - Read/write commands
  - Transferring data
  - CPU waits for I/O module to complete operation
  - Wastes CPU time
- Interrupt Driven I/O
  - Overcomes CPU waiting
  - No repeated CPU checking of device
  - I/O module interrupts when ready
- Direct Memory Access (DMA)
  - Additional Module (hardware) on bus
  - DMA controller takes over from CPU for I/O operations

Synchronizing the CPU and the Interface Chip

- The polling method:
  - for input -- the microprocessor checks a status bit of the interface chip to find out if the interface has received new data from the input device.
  - for output -- the microprocessor checks a status bit of the interface chip to find out if it can send new data to the interface chip.

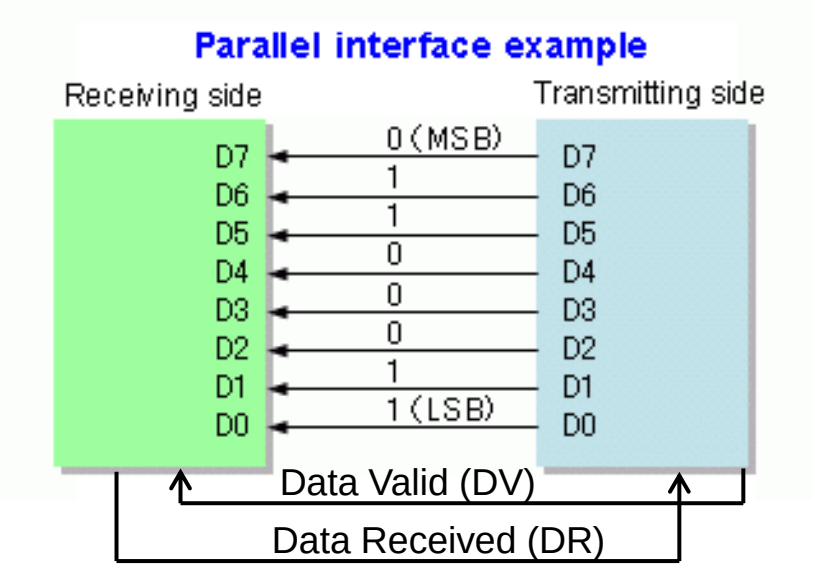
- The interrupt-driven method
  - for input -- the interface chip interrupts the microprocessor whenever it has received new data from the input device.
  - for output -- the interface chip interrupts the microprocessor whenever it can accept new data from the microprocessor.

CPU and I/O Interfaces

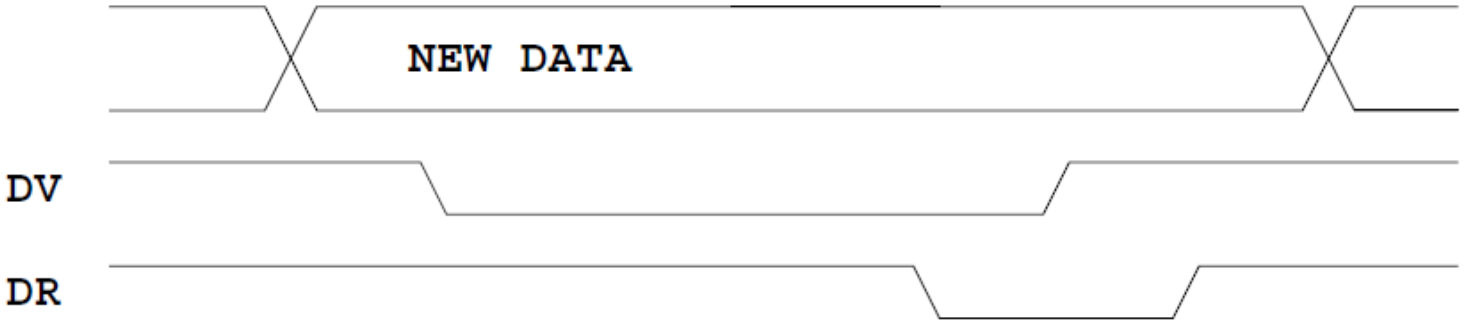


Synchronization of CPU and interface chip

- Transmitter initiated 2-wire Handshaking
  - In Parallel communication transmitter and receiver uses handshaking protocol
  - In short distance serial communication, handshaking can also be used

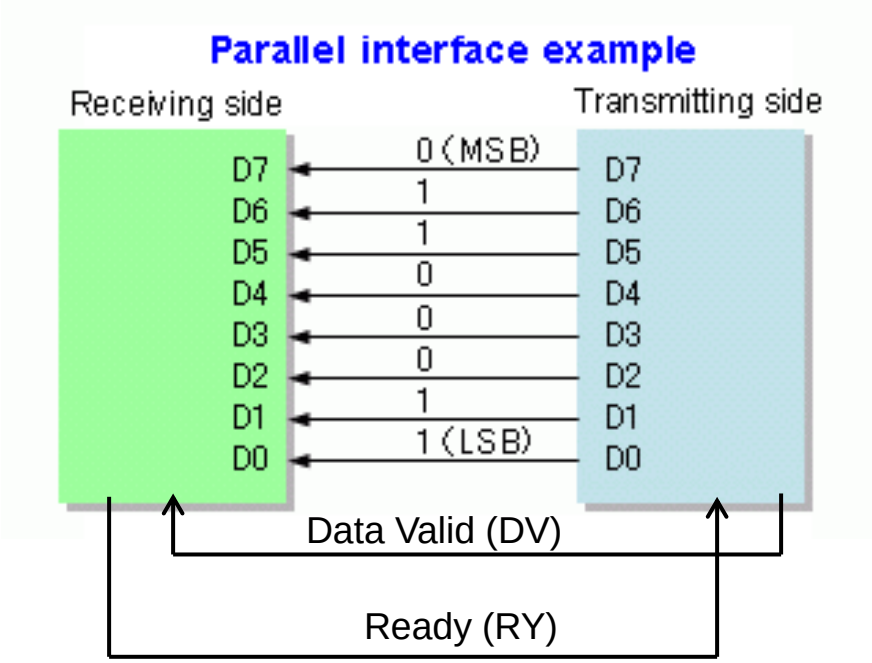


- Transmitter puts the data on the data lines and brings DV low to indicate new data is available
- When the receiver sees the new data is available, it reads the data, then brings DR low to say that it has read the data
- When the sending computer sees DR go low, it brings DV high
- When the receiving computer sees DV go high, it brings DR high
- Both computers are now ready for the next data transfer

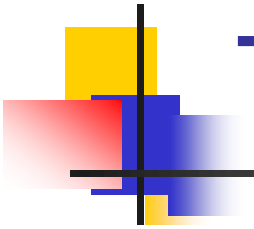


Synchronization of CPU and interface chip

- Receiver initiated 2-wire Handshaking
  - In Parallel communication transmitter and receiver uses handshaking protocol
  - In short distance serial communication, handshaking can also be used



- Receiver brings Ready (RY) to low to indicate it is ready to receive new data
- Transmitter places data on data lines and lowers Data Valid (DV)
- Receiver sees DV and accepts data, then brings RY high
- Transmitter sees RY high and raises DV to high
- Both computers are now ready for the next data transfer

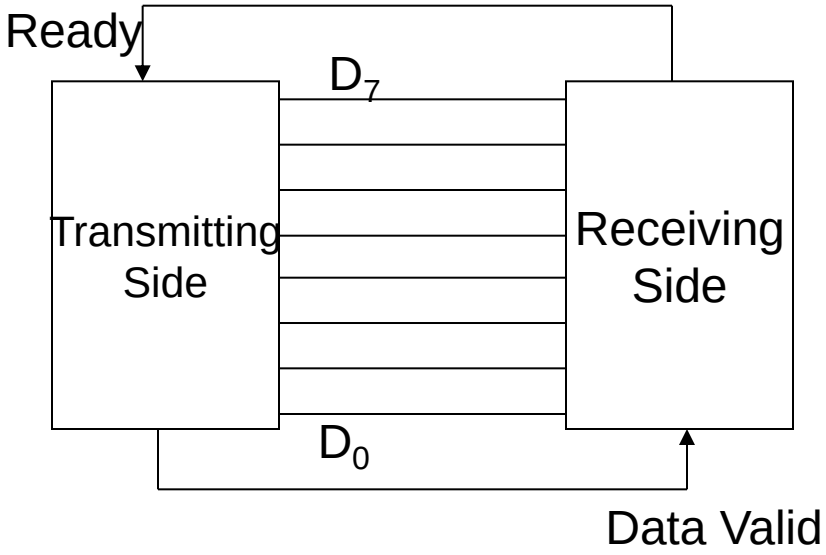


Topics

- I/O Interface
- I/O Transfer Synchronization
- Parallel Communication

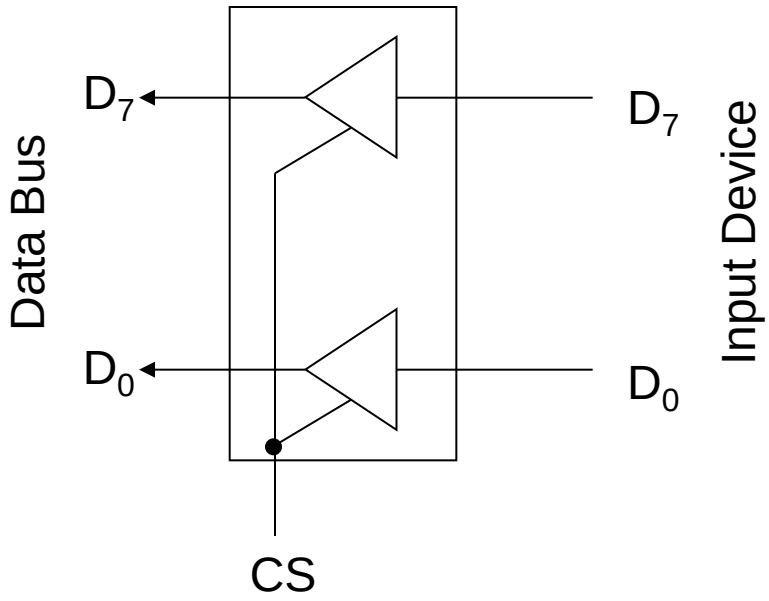
Parallel Communication

- Each bit is transferred along its own line and bits are transferred at the same time
- In addition to eight parallel data lines, other lines are used to read status information and send control signals.

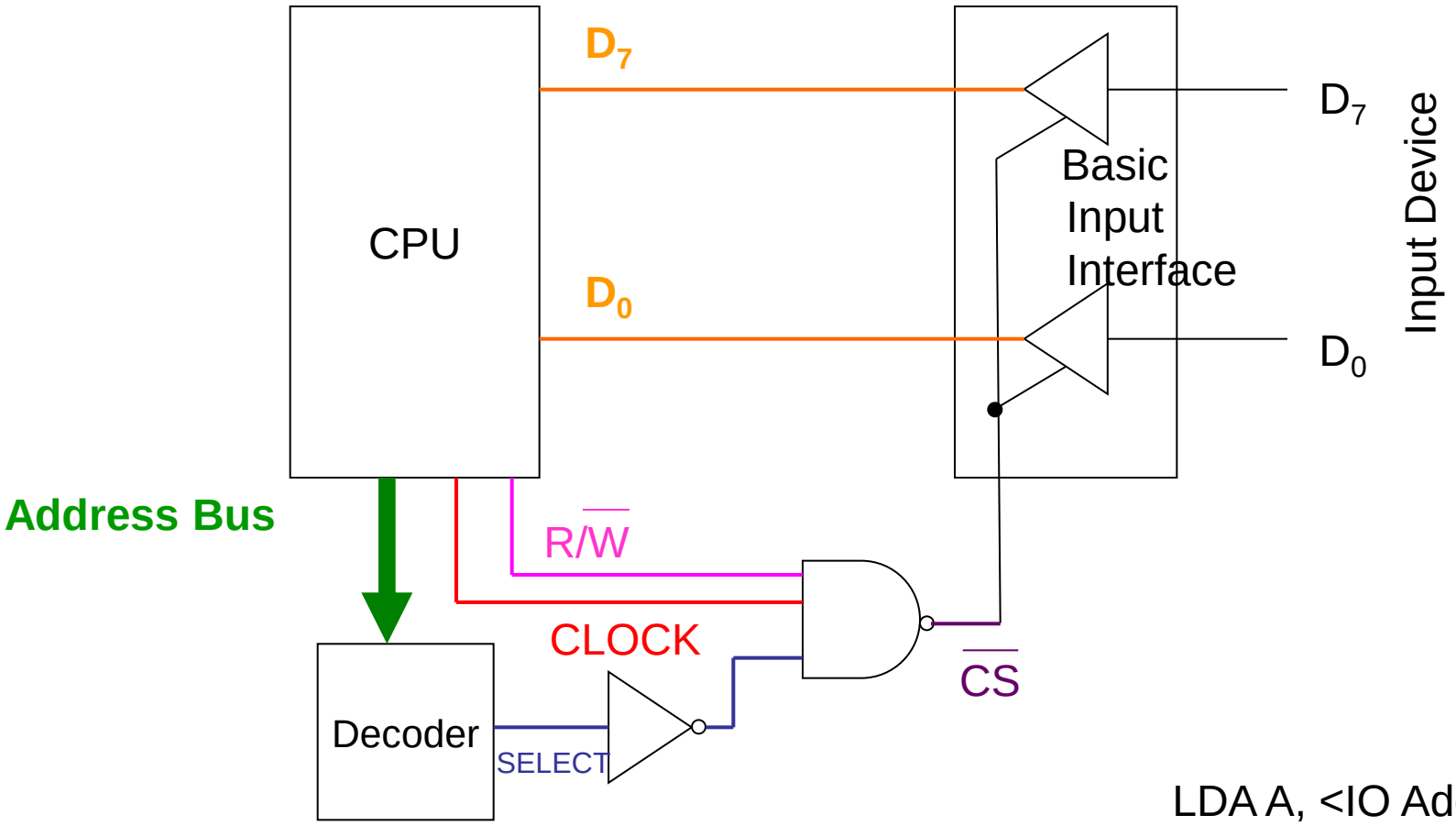


Basic Parallel Input Interface

- Tri-state buffers are used. (74LS244)
- Output pins connected to CPU Data Bus, Input pins connected to the Input device.



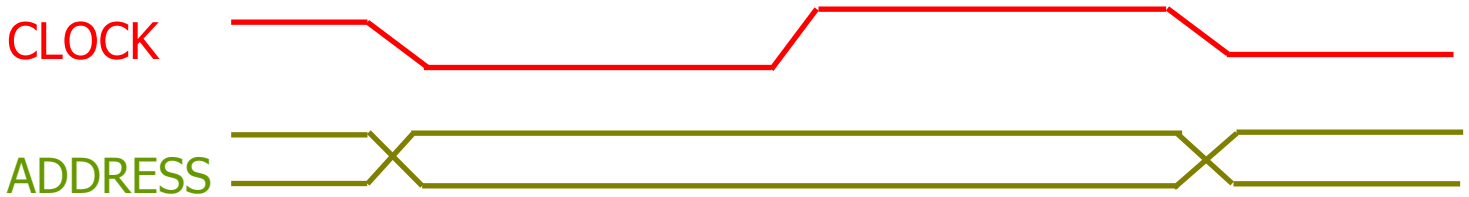
Basic Parallel Input Interface – CPU Connections



LDA A, <IO Address>

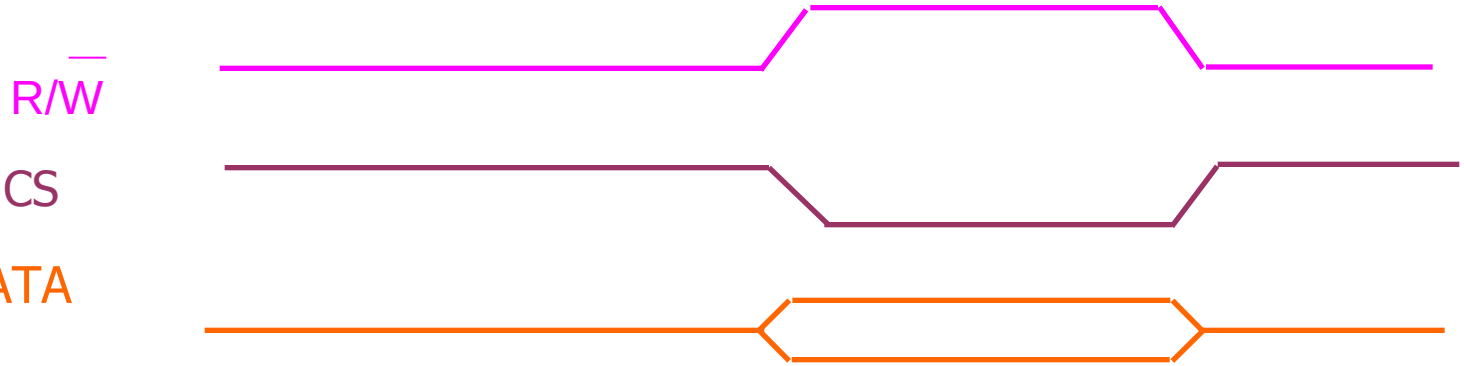
Basic Parallel Input Interface – CPU Control -

- Address placed on address bus at the falling edge of clock



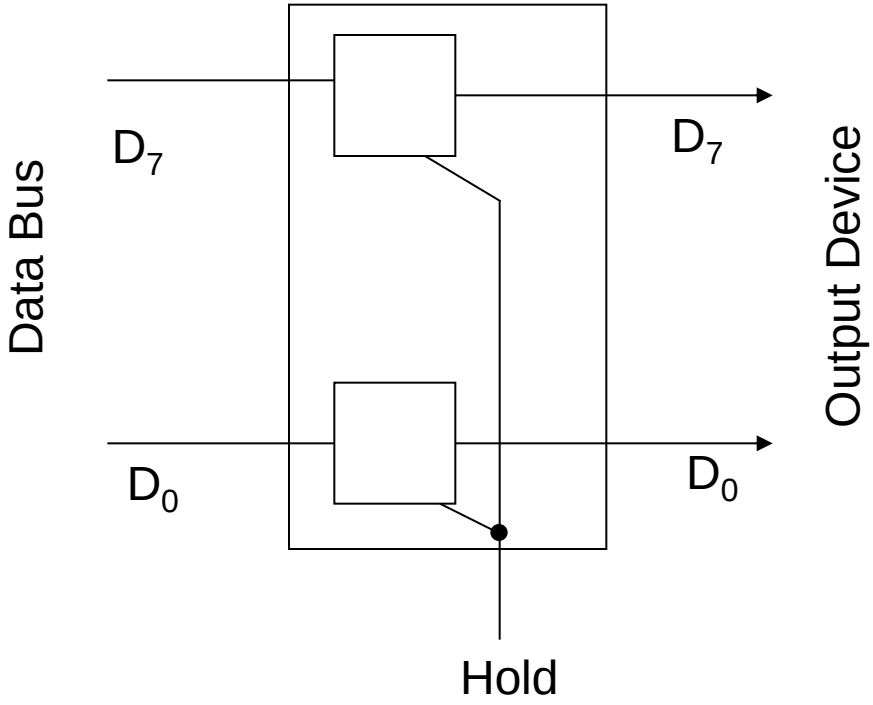
- Address decoded to form Select

- NAND {CLK, $\overline{\text{SELECT}}$, R/$\overline{\text{W}}$} to form CS

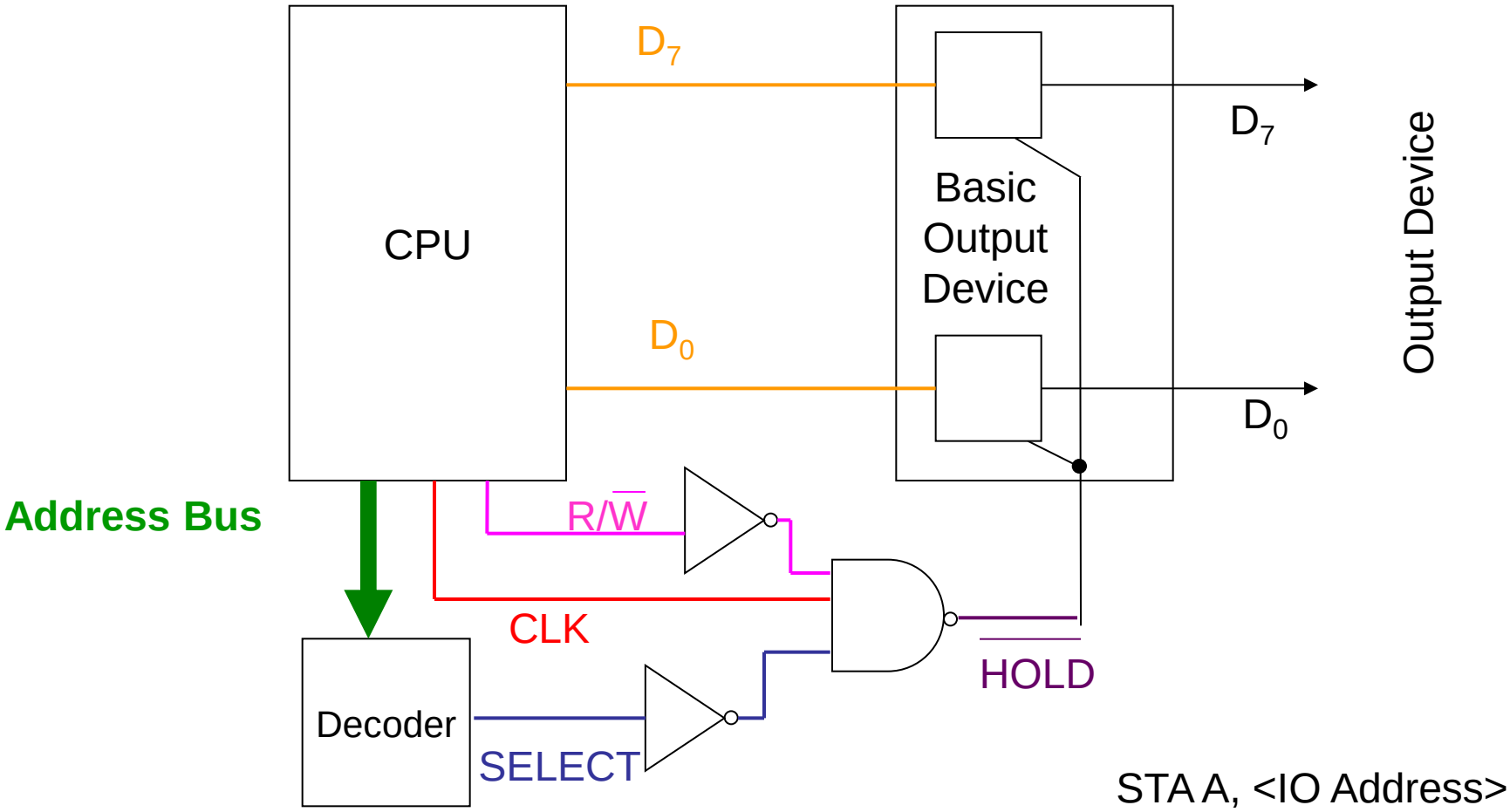


Basic Parallel Output Interface

- D Latches can be used (74LS374 Octal D Latch)
- Inputs of D latches are connected to the Data Bus. The outputs connected to the output device



Basic Parallel Output Interface – CPU Connections



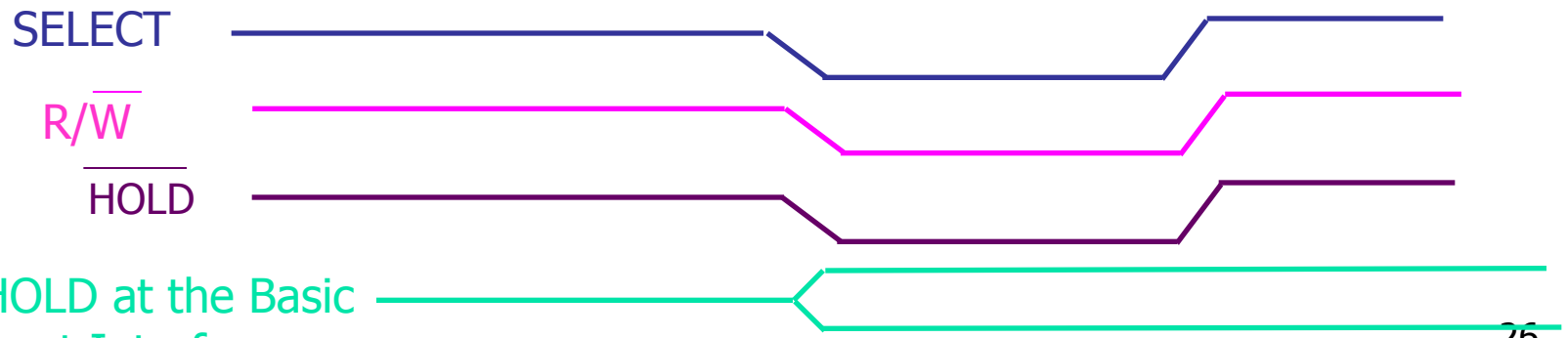
STA A, <IO Address>

Basic Parallel Output Interface – CPU Control

- Address placed on address bus at the falling edge of clock

- Data is placed on the data bus

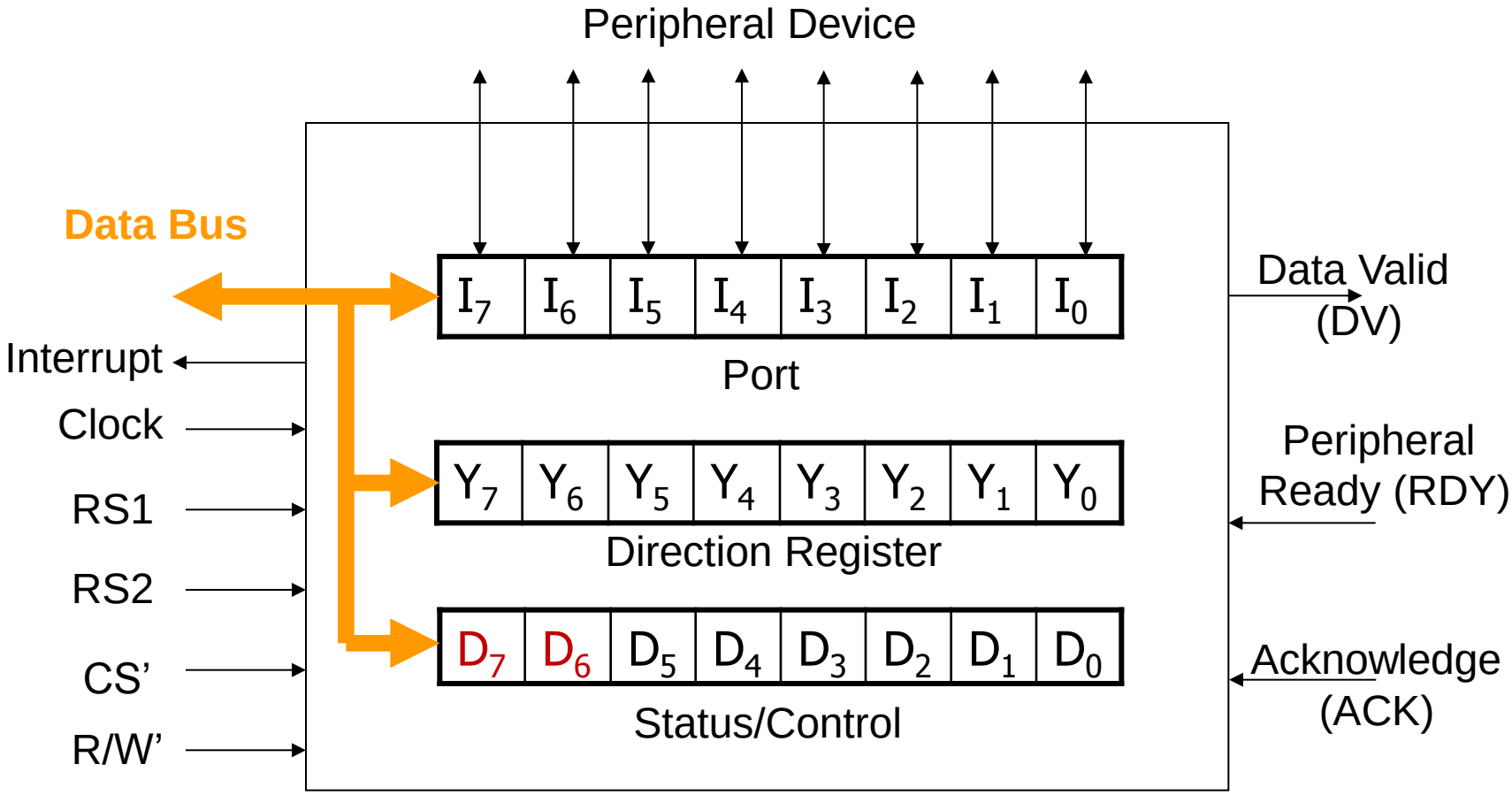
- Address decoded to form Select
- NAND {CLK, $\overline{\text{SELECT}}$, $\overline{\text{R/W}}$} to form HOLD



Peripheral Interface Adapter (EDU-PIA)

- Can be programmed to handle input or output data
- PIA contains
  - Port(s): Connects the PIA to peripheral devices. Each pin of the port can be conditioned as transmit (TX) or receive (RX)
  - Data direction register: Conditions the port pins as TX or RX
    - 1 : Pin is Transmitter
    - 0 : Pin is Receiver
  - Control / Status Register:
    - Status bits: Indicates the status of the handshaking bits
    - Control bits: Used to control handshaking signals and interrupt conditions

Peripheral Interface Adapter (EDU-PIA)



RS: Register Select
CS: Control Signal

Peripheral Interface Adapter (EDU-PIA)

- No consensus on the condition of raising the flag (1→0 or 0→1?)
- READY (RDY) Input
  - Indicates whether the peripheral device is ready
  - D7 Status Bit indicates RDY is received
    - Active High
  - D1 and D0 control bits determine whether an interrupt will be generated when RDY is received
  - D7 is cleared when Status/Control register is read

D_1D_0	Ready (RDY) Input	Interrupt Output
0 0	1 → 0 => D_7=1	High, No interrupt
0 1	0 → 1 => D_7=1	High, No interrupt
1 0	1 → 0 => D_7=1	Hi->Low->Hi, Interrupt
1 1	0 → 1 => D_7=1	Hi->Low->Hi, Interrupt

Peripheral Interface Adapter (EDU-PIA)

- No consensus on the condition of raising the flag (1→0 or 0→1?)
- ACKNOWLEDGE (ACK) Input
  - Indicates whether the peripheral device received the data
  - D6 Status Bit indicates ACK is received
    - Active High
  - D3 and D2 control bits determine whether an interrupt will be generated when ACK is received
  - D6 is cleared when Status/Control register is read

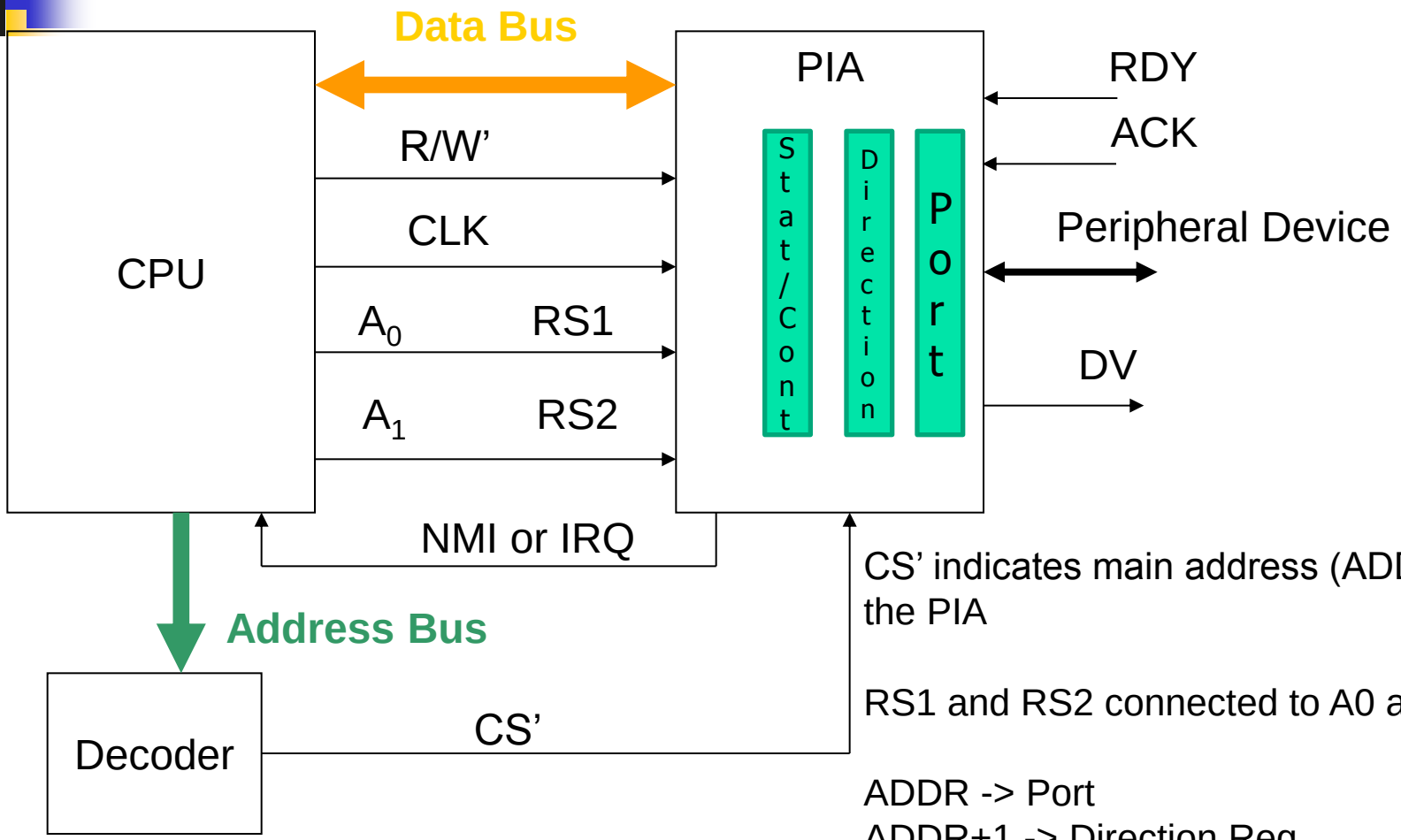
D_3D_2	Acknowledge (ACK) Input	Interrupt Output
0 0	1 → 0 => D_6=1	High, No interrupt
0 1	0 → 1 => D_6=1	High, No interrupt
1 0	1 → 0 => D_6=1	Hi->Low->Hi, Interrupt
1 1	0 → 1 => D_6=1	Hi->Low->Hi, Interrupt

Peripheral Interface Adapter (EDU-PIA)

- No consensus on the handshake conditions
- DATA VALID (DV) Output
  - Indicates valid data is at the Port
  - D5 and D4 control bits determine the DV conditions

D_5	D_4	DATA VALID (DV)
0	0	DV is reset (low)
0	1	DV is set (high)
1	0	1 after the data is loaded on the port
1	1	0 after the data is loaded on the port

EDU-PIA / CPU Connections

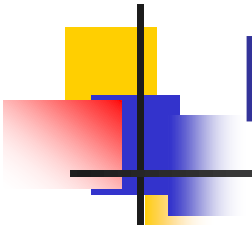


CS' indicates main address (ADDR) of the PIA

RS1 and RS2 connected to A0 and A1

ADDR -> Port	$8080
ADDR+1 -> Direction Reg.	$8081
ADDR+2 -> Status/Control Reg.	$8082

NMI:Non-Maskable Interrupt

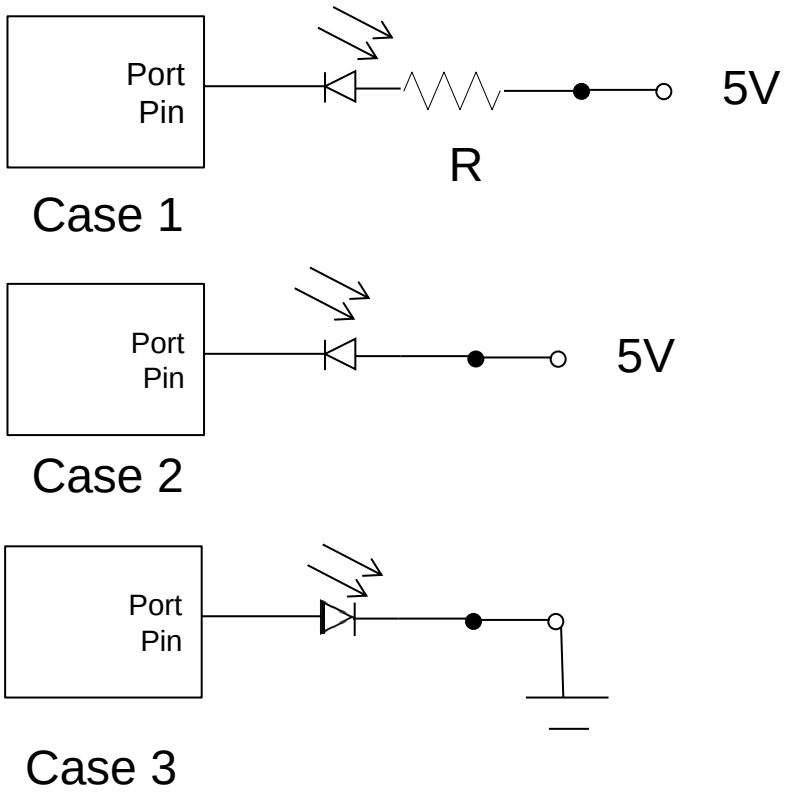


EDU-PIA / CPU Connections

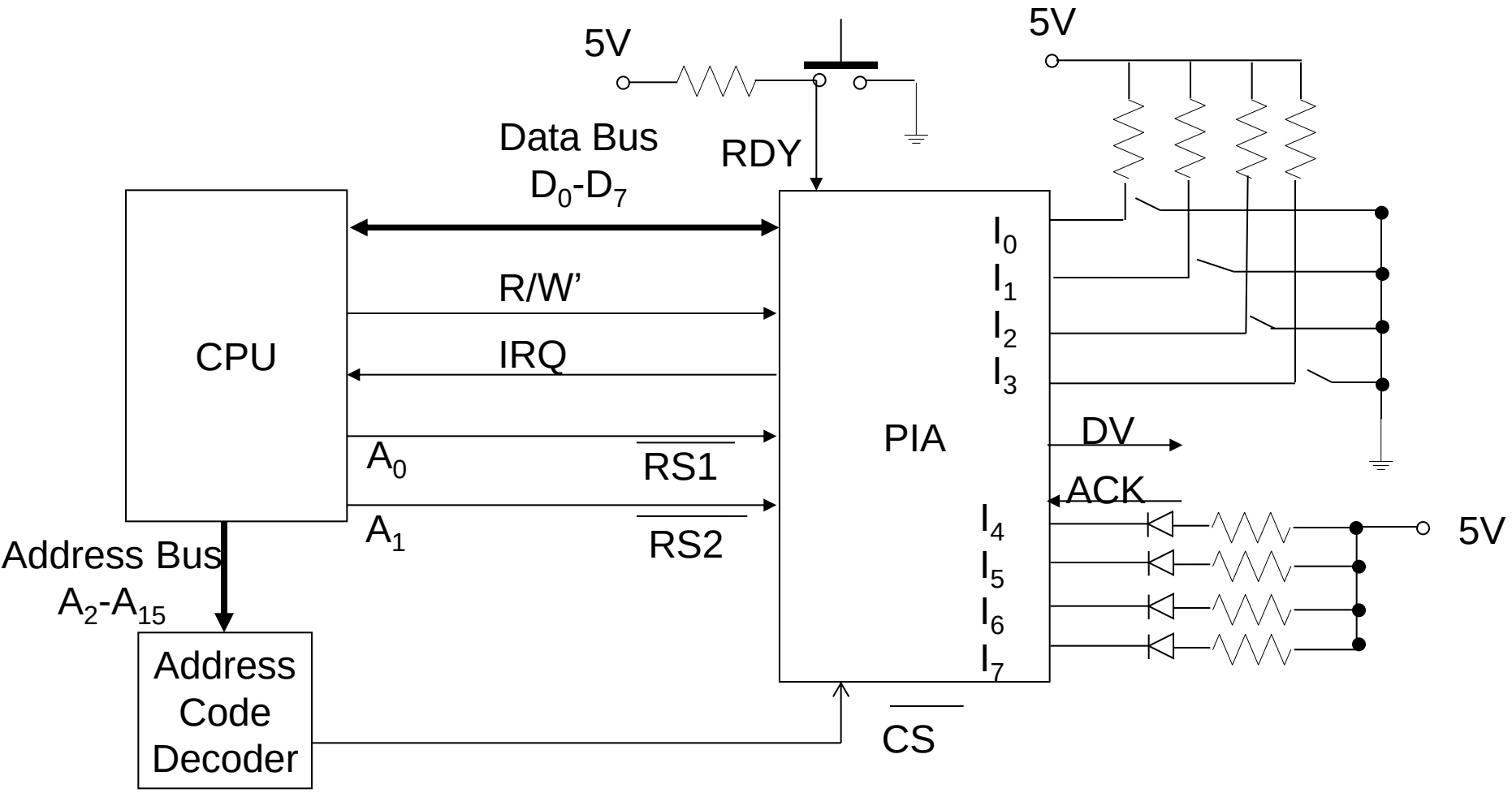
- Example I: The EDU-PIA is connected to an 8-bit microprocessor with 16-line address bus. The main address of the PIA is $A0A0. The first four pins of the PIA are connected to four switches. The remaining four bits of the PIA are connected to LEDs.
  - Design a system that will illuminate the LEDs depending on the switch positions. The LEDs will illuminate after the user arranges the switches and presses a button.

Simple Output Device - LED

- Case-1
  - LED is ON for an output of zero
  - Most LEDs drop 1.7 to 2.5 volts and need about 10mA
  - Current is (5-2)/R
- Case-2
  - Too much current
  - Failure of Port or LED
- Case-3
  - Not enough drive (1mA)
  - LED too dim

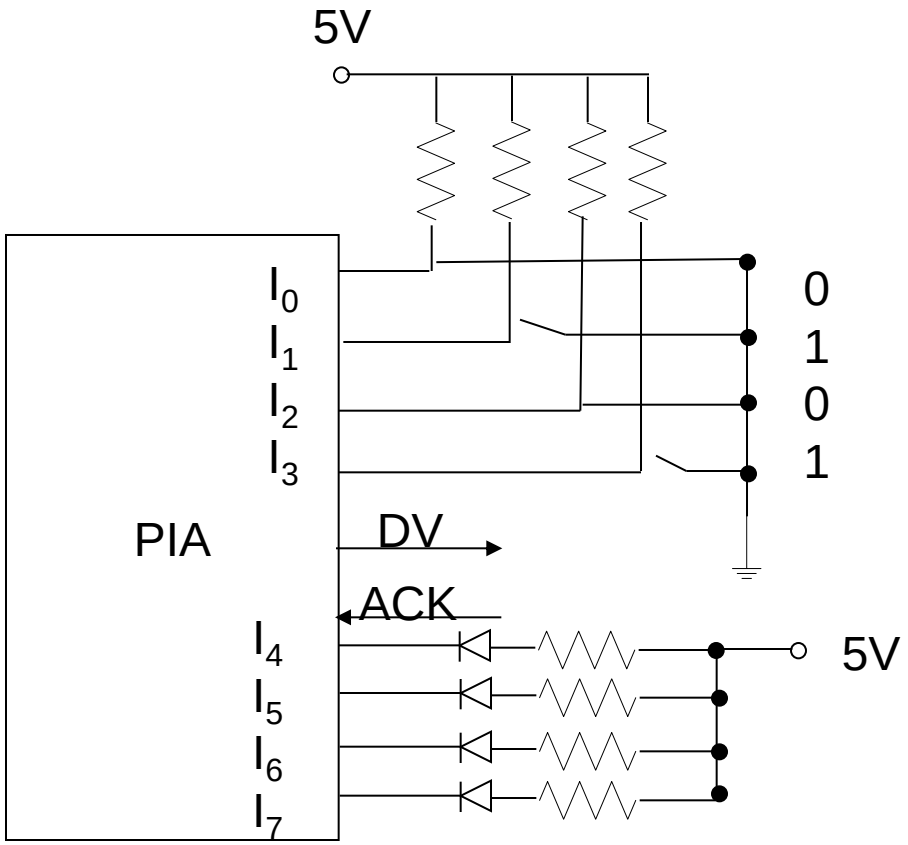


Example-I


5V
5V
Data Bus
D_0-D_7
RDY
CPU
R/W'
IRQ
A_0
A_1
RS1
RS2
PIA
I_0
I_1
I_2
I_3
I_4
I_5
I_6
I_7
DV
ACK
5V
Address Bus
A_2-A_{15}
Address Code Decoder
CS

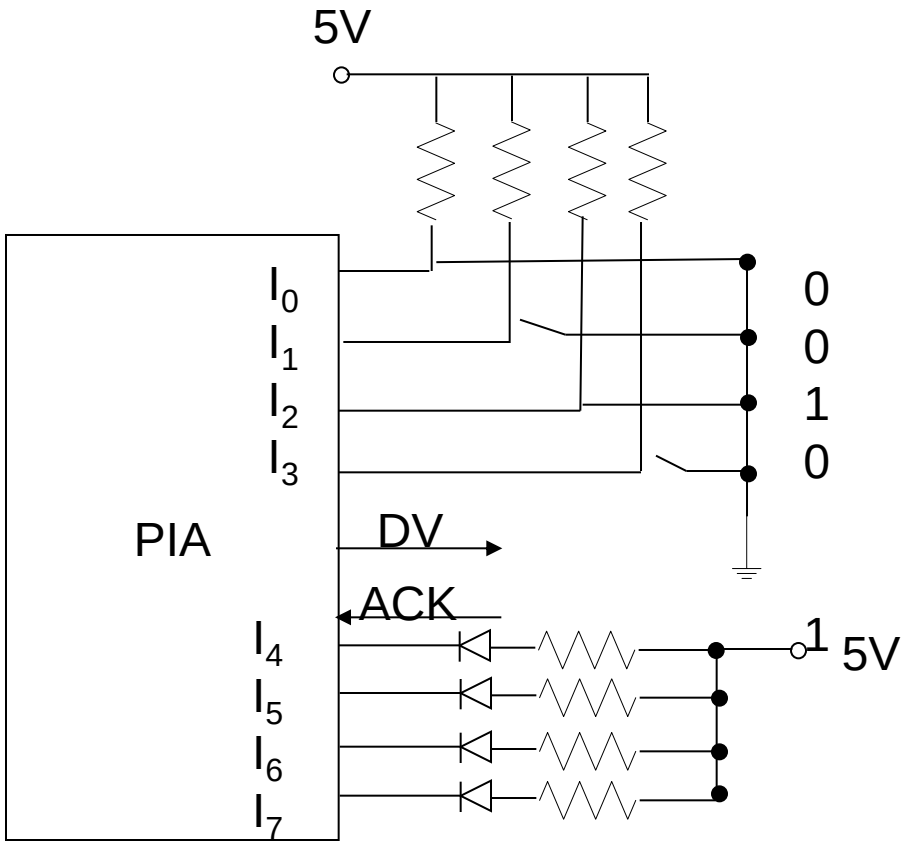
Example-I

```
PORT              EQ       $A0A0
DIRECT            EQ       $A0A1
STATCON           EQ       $A0A2


START    LDA      A,$F0
         STA      A,DIRECT
         LDA      A,$00
         STA      A,STATCON
REW      LDA      A,<STATCON>
         TST      A,$80
         BEQ      REW
         LDA      A,<PORT>
```



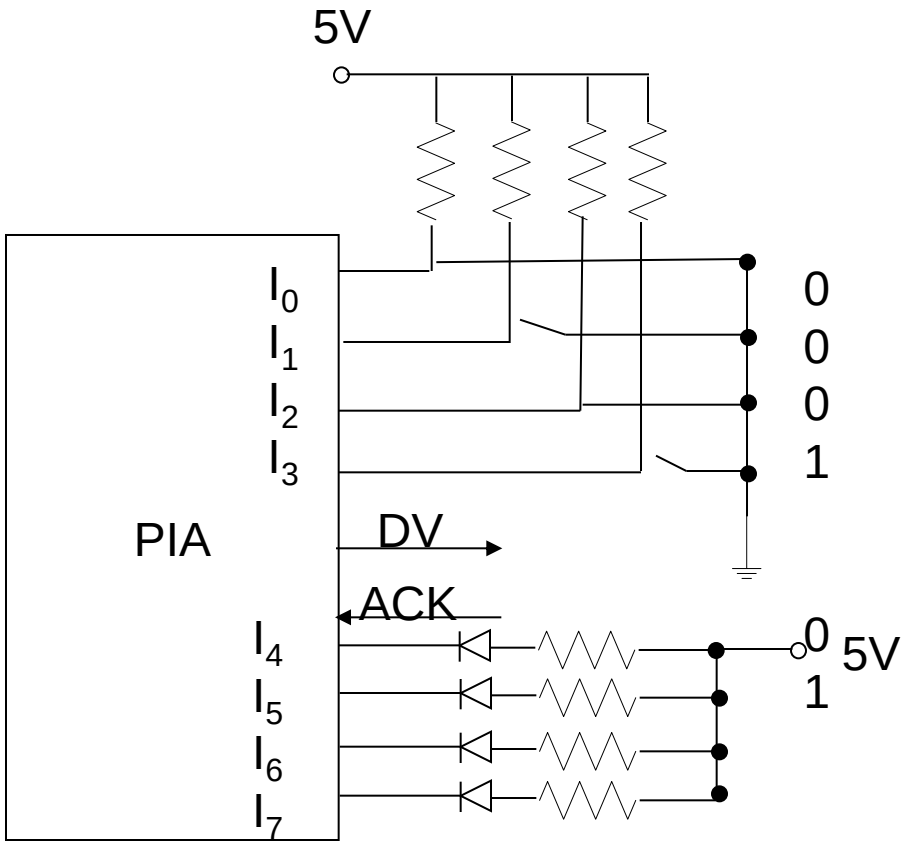
Example-I

```
PORT            EQ      $A0A0
DIRECT          EQ      $A0A1
STATCON         EQ      $A0A2


START   LDA     A,$F0
        STA     A,DIRECT
        LDA     A,$00
        STA     A,STATCON
REW     LDA     A,<STATCON>
        TST     A,$80
        BEQ     REW
        LDA     A,<PORT>
        SHL     A
```



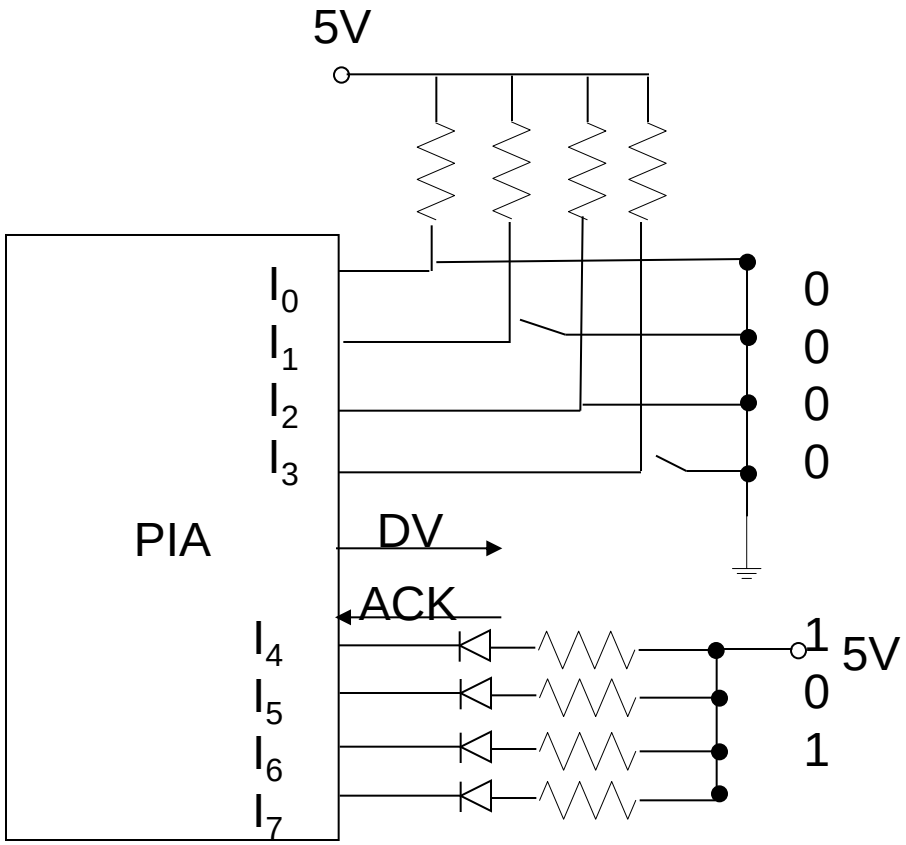
Example-I

```
PORT              EQ       $A0A0
DIRECT            EQ       $A0A1
STATCON           EQ       $A0A2


START    LDA      A,$F0
         STA      A,DIRECT
         LDA      A,$00
         STA      A,STATCON
REW      LDA      A,<STATCON>
         TST      A,$80
         BEQ      REW
         LDA      A,<PORT>
         SHL      A
         SHL      A
```



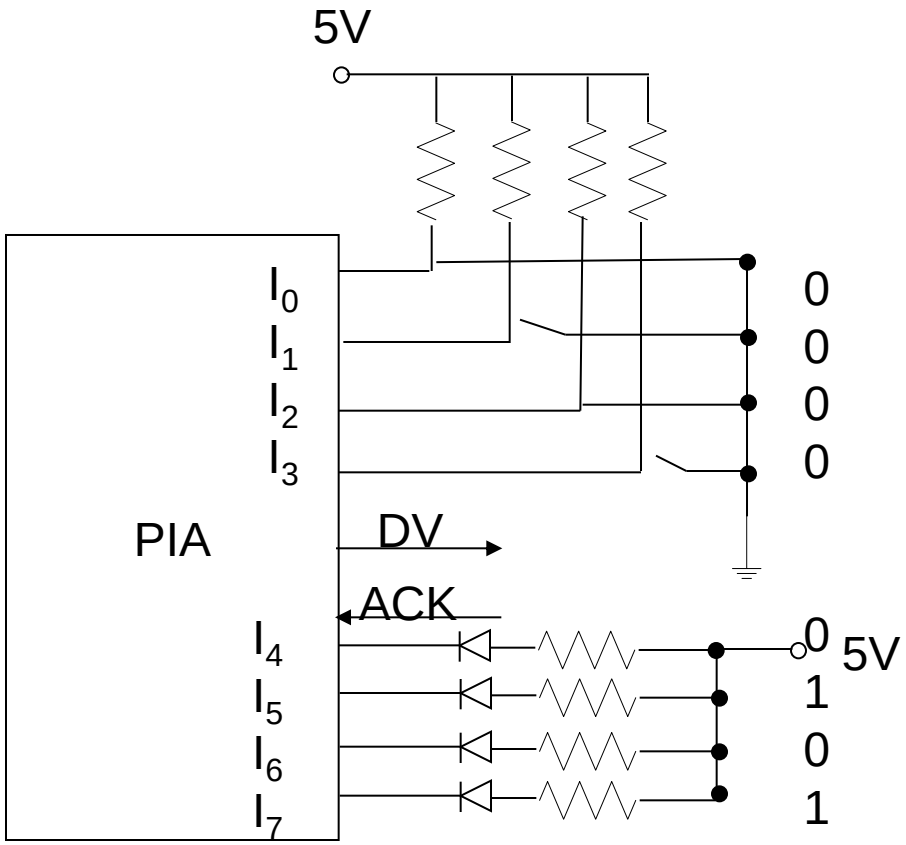
Example-I

```
PORT              EQ       $A0A0
DIRECT            EQ       $A0A1
STATCON           EQ       $A0A2


START    LDA      A,$F0
         STA      A,DIRECT
         LDA      A,$00
         STA      A,STATCON
REW      LDA      A,<STATCON>
         TST      A,$80
         BEQ      REW
         LDA      A,<PORT>
         SHL      A
         SHL      A
         SHL      A
```



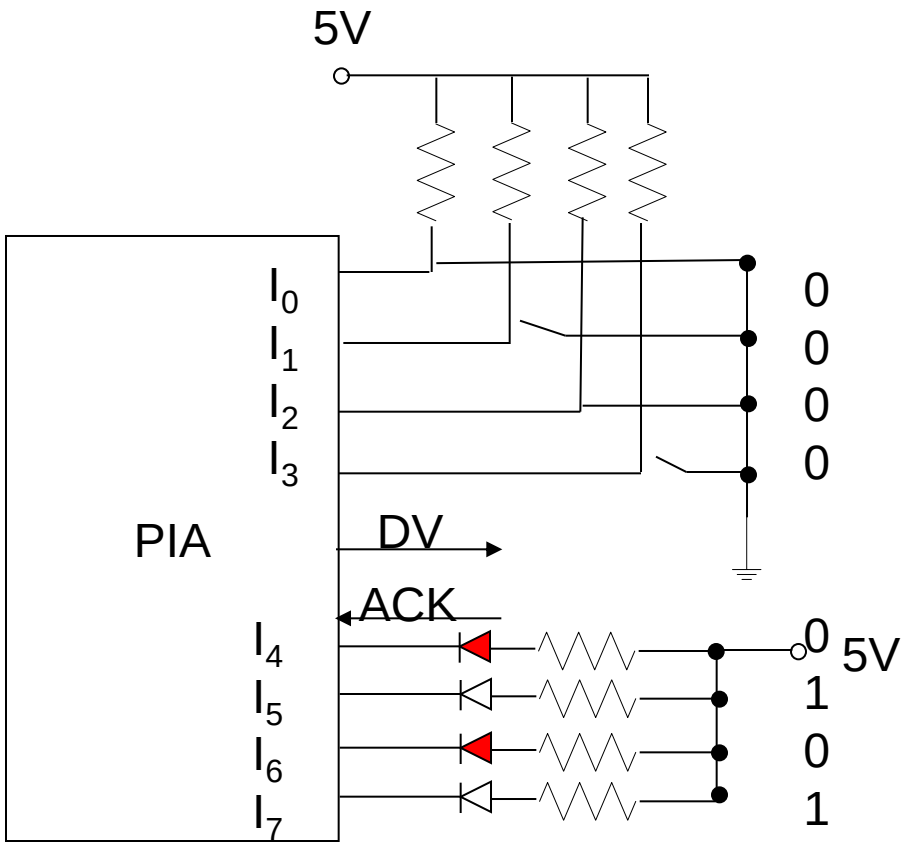
Example-I

```
PORT              EQ       $A0A0
DIRECT            EQ       $A0A1
STATCON           EQ       $A0A2


START    LDA      A,$F0
         STA      A,DIRECT
         LDA      A,$00
         STA      A,STATCON
REW      LDA      A,<STATCON>
         TST      A,$80
         BEQ      REW
         LDA      A,<PORT>
         SHL      A
         SHL      A
         SHL      A
         SHL      A
```



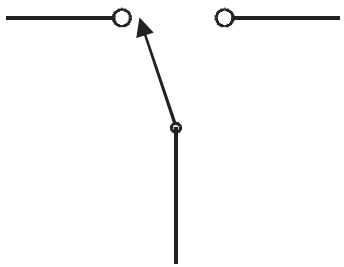
Example-I

```
PORT              EQ       $A0A0
DIRECT            EQ       $A0A1
STATCON           EQ       $A0A2


START    LDA      A,$F0
         STA      A,DIRECT
         LDA      A,$00
         STA      A,STATCON
REW      LDA      A,<STATCON>
         TST      A,$80
         BEQ      REW
         LDA      A,<PORT>
         SHL      A
         SHL      A
         SHL      A
         SHL      A
         STA      A,<PORT>
         BR       REW
```



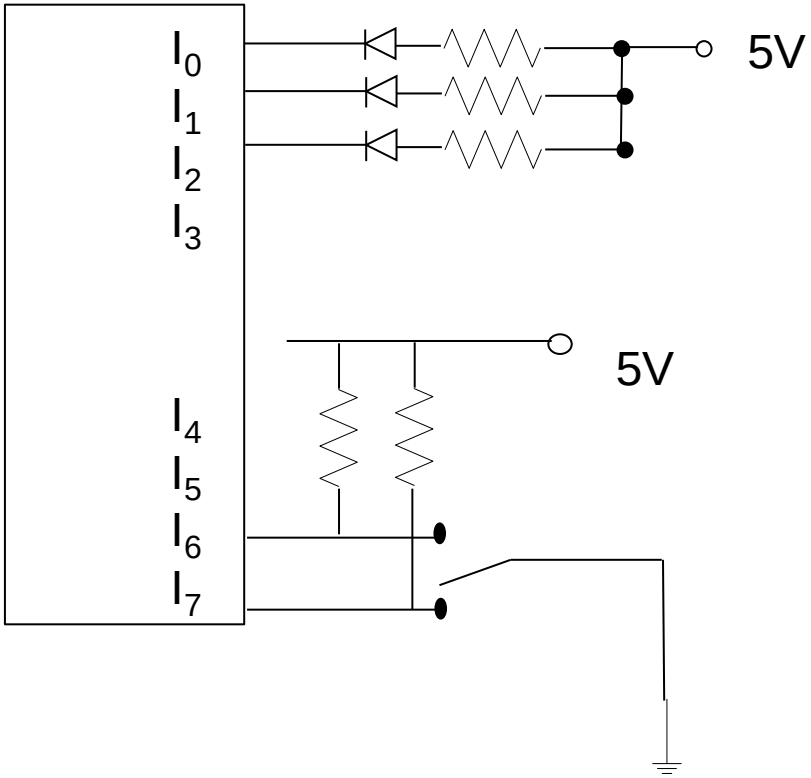
Example-II

- The first three pins (I0, I1 ve I2) of the PIA are connected to three LEDs, and the last two pins (I6, I7) are connected to a switch as shown in the figure. This switch connects etiher I6 or I7 to ground.



  - Design and draw the hardware clearly showing PIA, switch, LEDs and resistors.
  - If the switch connects I6 to ground, first the LED connected to I0 will light for 0.5 s. Afterwards the LED connected to I1 will light 0.5 s., then the LED connected to I2 will light for 0.5 s. After 0.5 s. all the LEDs will fade out. If the switch connects I7 to ground, then the same procedure will occur beginning to light the LED connected to I2. Write the program to perform this operation.

Example-II



Example-II

```
START   LDA     SP,$FFFF
        BSR     COND
        CLR     E
REW     LDA     A,<PORT>
        TST     A,$80
        BEQ     K7
        TST     A,$40
        BEQ     K6
        BR      REW
K7      LDA     B,$03
        LDA     A,$03
AGN1    STA     A,<PORT>
        BSR     DELAY
        SHR     A
        DEC     B
        BNEQ    AGN1
        BSR     FDO
        BR      REW

K6      LDA     B,$03
        LDA     A,$FE
AGN2    STA     A,<PORT>
        BSR     DELAY
        SHL     A
        DEC     B
        BNEQ    AGN2
        BSR     FDO
        BR      REW

COND    LDA     A,$07
        STA     A,<DIRECT>
        LDA     A,$00
        STA     A,<STAT/CON>
        RTS

FDO     LDA     A,$FF
        STA     A,<PORT>
        RTS
```

Example-II

```
DELAY   LDA     IX,$F422        (2 µs)
DELAY1  DEC     IX              (2 µs)
        BNEQ    DELAY1          (2 µs)
        LDA     IX,$F422
DELAY2  DEC     IX
        BNEQ    DELAY2
        RTS
```

0,5s=500.000 µs

500000-12=499.988 µs

Time in one iteration: 4 µs

499.988/4=124.997 iterations

124.997/2= 62.498 µs => $F422

Parallel vs. Serial Connection

- Before the development of high-speed serial technologies, the choice of parallel links over serial links was driven by these factors:
  - Speed: Superficially, the speed of a parallel data link is equal to the number of bits sent at one time times the bit rate of each individual path
  - Cable length: Crosstalk creates interference between the parallel lines, and the effect worsens with the length of the communication link. This places an upper limit on the length of a parallel data connection that is usually shorter than a serial connection.
  - Complexity: Parallel data links are easily implemented in hardware, making them a logical choice. Creating a parallel port in a computer system is relatively simple, requiring only a latch to copy data onto a data bus. In contrast, most serial communication must first be converted back into parallel form by a universal asynchronous receiver/transmitter (UART) before they may be directly connected to a data bus.
- The decreasing cost of integrated circuits, combined with greater consumer demand for speed and cable length, has led to parallel communication links becoming deprecated in favor of serial links; for example, IEEE 1284 printer ports vs. USB, Parallel ATA vs. Serial ATA, and SCSI vs. FireWire.