MICROPROCESSOR SYSTEMS

# Lecture 8

Parallel Communication

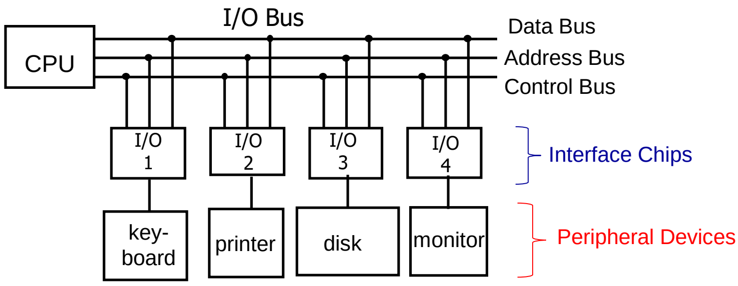


## Topics

- $\boxed{\text{Input/Output Interface}}$
- I/O Transfer Synchronization
- Parallel Communication

# Input/Output Interfaces

- **Peripheral Devices** are I/O devices that exchange data with a CPU.
  - Examples : Switch, push-button, light-emitting diode (LED), monitor, LCD screen, printer, mouse, keyboard, disk drive, sensor, motor, audio, etc.
- **Interface Chips** are used to resolve the differences between CPU and I/O devices.




3

# I/O Interfacing

- An interface chip resolves the differences between the CPU and a peripheral device.
- Data codes and formats in peripherals may be different.
- A synchronization mechanism is needed.
  Data transfer rate of peripherals are usually slower than CPU.
- Communication between CPU and Interface Chip:
  Always parallel.
- Communication between Interface Chip and I/O Device:
  Can be either parallel or serial.

4

# I/O Interfacing Methods

- **Isolated I/O Method**
  - Separate memory and I/O address spaces are used.
  - There are distinct input and output instructions.
    Example assembly language instructions : **IN , OUT**

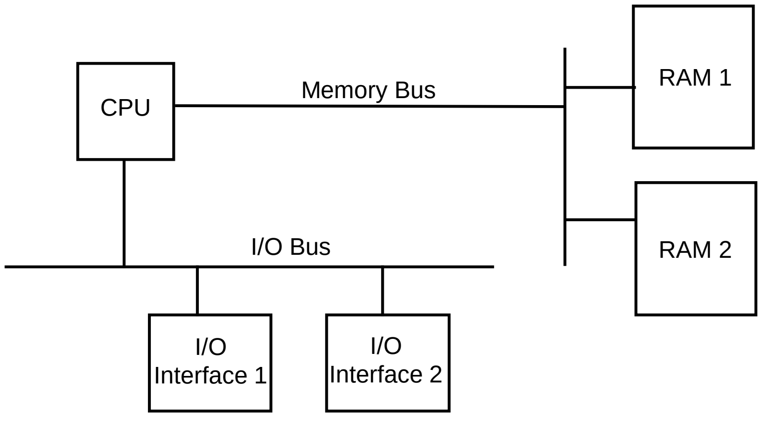
- **Memory Mapped I/O Method**
  - Memory and I/O addresses share common address space.
  - There are no specific input/output instructions.
  - Usual memory load/store instructions are used.
    Example assembly language instructions : **LDA , STA**
  - Flexibility in handling I/O operations.

5

# Isolated I/O Interfacing

Separate Buses are used for Input-Output Interfaces and Memory.

- Memory Bus : Address & Data & Control
- Input-Output Bus : Address & Data & Control

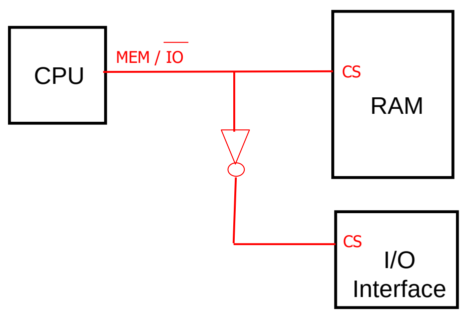



6

# Isolated I/O Interfacing

MEM / $\overline{IO}$ control signal is used as Memory chip selection and Input-Output Interface chip selection.

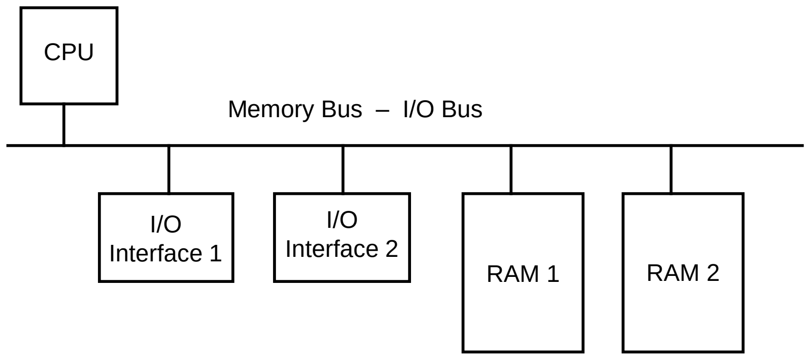
- MEM / IO′ = 1 means Memory chip is selected
- MEM / IO′ = 0 means Input-Output chip is selected





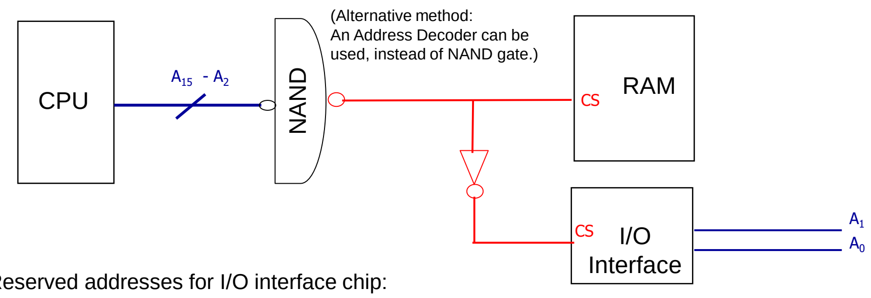

# Memory Mapped I/O Interfacing

A Common Bus is used for I/O Interfaces and Memory.



# Example: Memory Mapped I/O Interfacing

- Example: 4 memory addresses are reserved (dedicated) for Input-Output register addresses.
- Suppose the memory addresses from \$0000 to \$0003 are reserved for I/O registers.
- The NAND gate can be used on address lines $A_{15}$ - $A_2$ for chip selections (RAM or I/O).
- For I/O register selections within the I/O interface chip, $A_0$ and $A_1$ lines can be used.




Reserved addresses for I/O interface chip:

| Address (Hex) | $A_{15}$ | $A_{14}$ | $A_{13}$ | $A_{12}$ | $A_{11}$ | $A_{10}$ | $A_9$ | $A_8$ | $A_7$ | $A_6$ | $A_5$ | $A_4$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| \$0000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$0001 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| \$0002 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| \$0003 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

| Address Map | Used by which chip |
|---|---|
| \$0000 - \$0003 | I/O |
| \$0004 - \$FFFF | RAM |

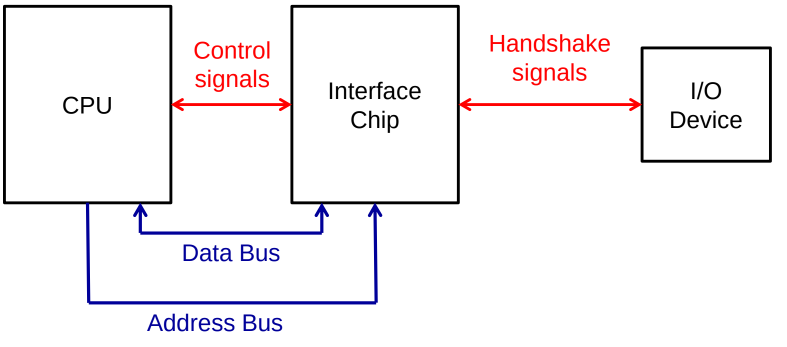


# Topics

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

# I/O Transfer Synchronization

- The tasks of the interface chip are :
  - Synchronizing data transfer between <u>CPU and Interface chip</u> (Control signals).
  - Also between <u>Interface chip and I/O device</u> (Handshaking signals).






# CPU and I/O Interfaces

- **Programmed I/O (Polling method)**
  - CPU checks device status in a loop
  - CPU waits for I/O module to complete operation
  - Wastes CPU time
- **Interrupt Driven I/O**
  - Overcomes CPU unnecessary waiting
  - No repeated CPU checking of device
  - I/O module interrupts when ready
- **Direct Memory Access (DMA)**
  - Requires additional hardware module on bus
  - DMA controller takes over from CPU, for I/O operations

# Programmed I/O (Polling method)

- Input Operations :
  **Microprocessor** checks a status bit of the interface chip, to find out whether the interface **has received** new data from input device.

- Output Operations :
  **Microprocessor** checks a status bit of the interface chip, to find out whether it **can send** new data to interface chip.



# Interrupt Driven I/O

- Input Operations :
  **Interface chip** interrupts the microprocessor, whenever it **has received** new data from input device.

- Output Operations :
  **Interface chip** interrupts the microprocessor, whenever it **can accept** new data from microprocessor.

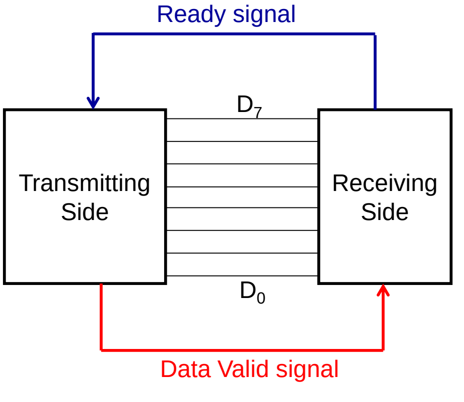
## Topics

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

15

## Parallel Communication

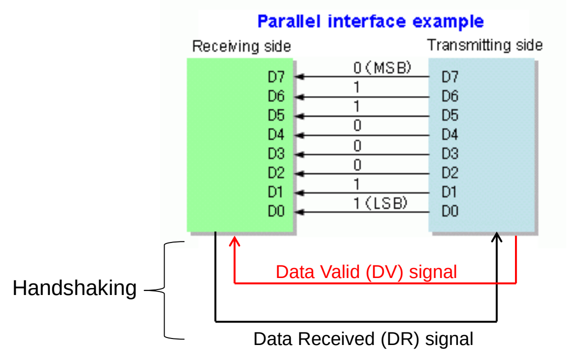
- In parallel communication, all bits are transferred at the same time.
- Each bit is transferred along its own line.
- In addition to eight parallel data lines, other lines are used to read status information and send control signals.




16

# Example: Synchronization of two parallel connected computers

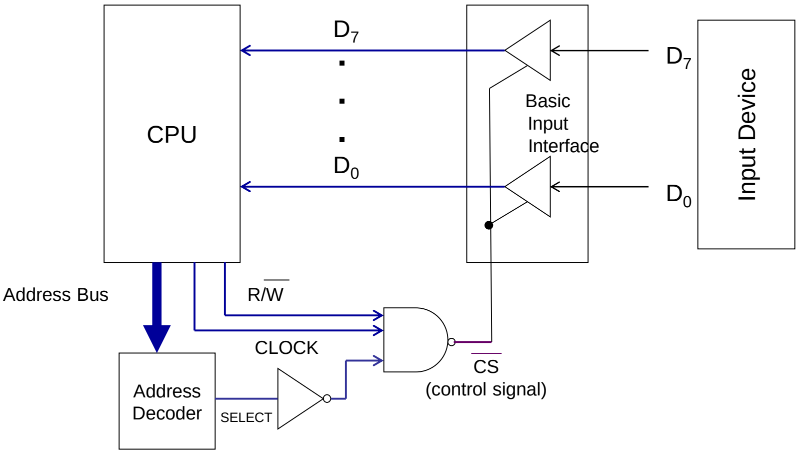
- **Transmitter** and **receiver** interfaces use handshaking protocol.
- **Transmitter** initiates 2-wire handshaking
  - DV low indicates new data is available.
  - DR low indicates that receiver has read the data.




17

# Basic Parallel Input Interface

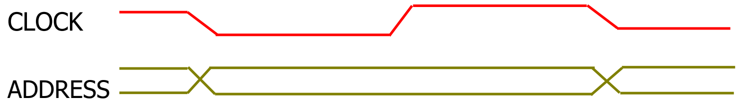
- **Tri-state buffers** are used inside the interface chip. (Example chip: 74LS244)
- Output pins of **interface** are connected to **Data Bus** of CPU (as input of CPU).
- Input pins of **interface** are connected to an input device.




18

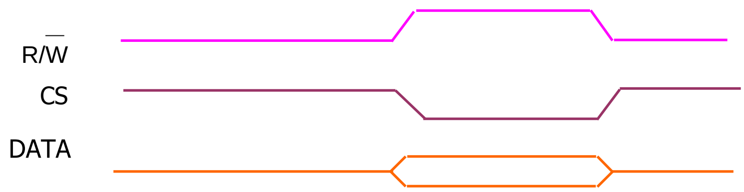
# Input Interface Timing Diagram

1) Address placed on address bus at the falling edge of clock




2) Address decoded, in order to form the Select signal

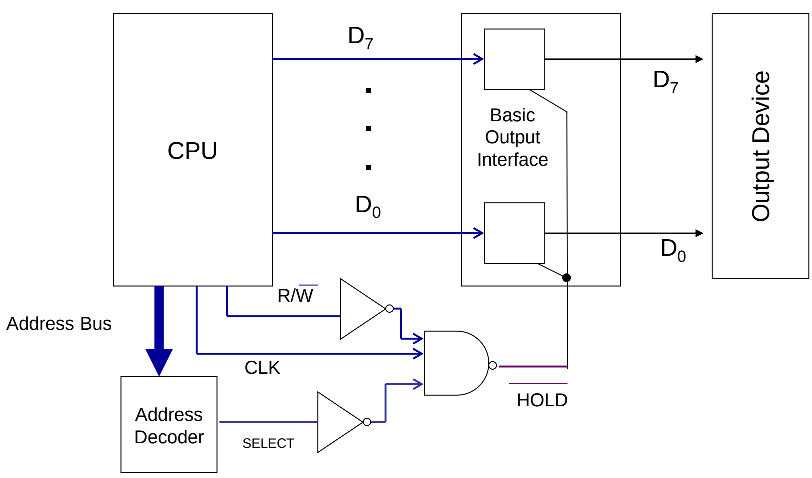

3) NAND {CLK, $\overline{\text{SELECT}}$, R/$\overline{\text{W}}$} applied, in order to form CS signal





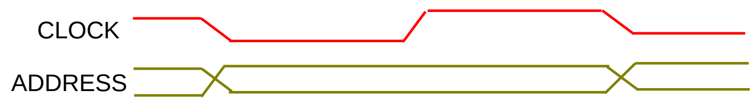

# Basic Parallel Output Interface

- **D Latches** are used in interface (Example chip: 74LS374 D Latch).
- Inputs of **interface** are connected to the **Data Bus** of CPU.
- Outputs of **interface** are connected to an output device.



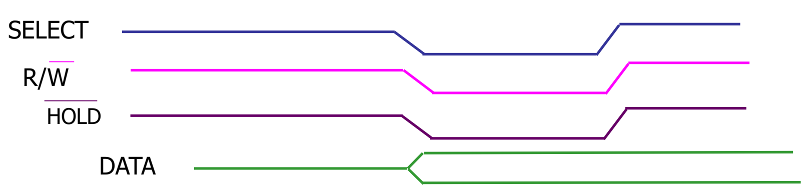
# Output Interface Timing Diagram

1) Address placed on address bus at the falling edge of clock




2) Data is placed on the data bus


3) Address decoded, in order to form the Select signal
4) NAND {CLK, $\overline{\text{SELECT}}$, R/$\overline{\text{W}}$} applied, in order to form $\overline{\text{HOLD}}$

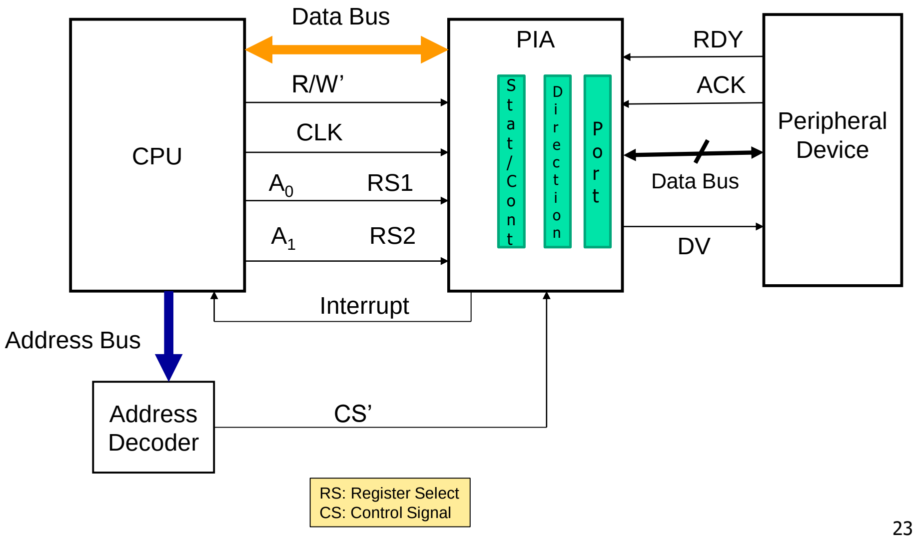





# Peripheral Interface Adapter (PIA)

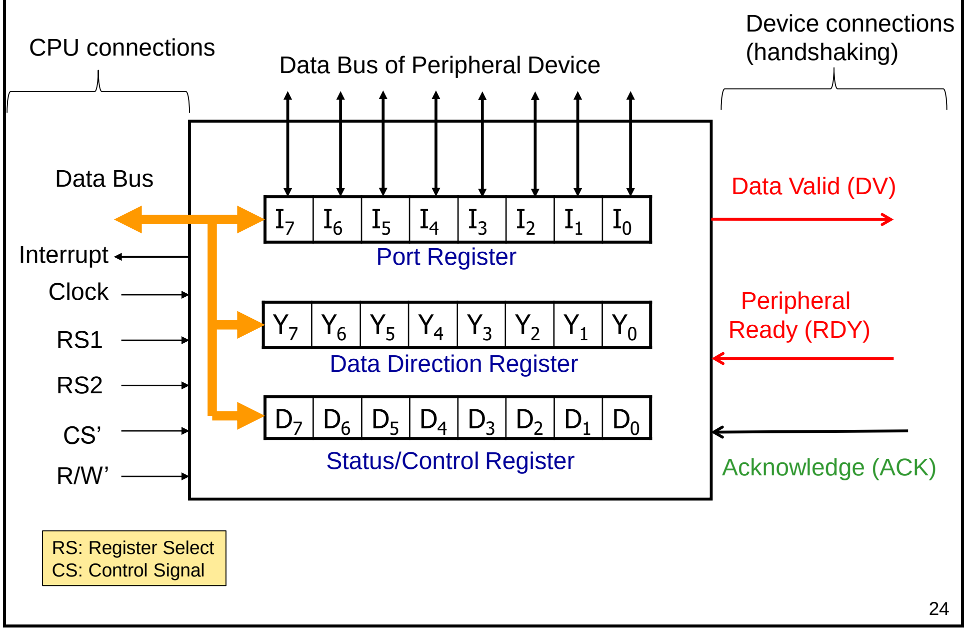
- PIA is a parallel interface chip that can be programmed to handle both input or output data.
- PIA contains the following 3 registers:
  - **Port Register(s):** Connects the PIA to peripheral devices. Each pin (bit) of the port can be conditioned (configured) as *transmit (TX)* or *receive (RX)*
  - **Data Direction Register:** Conditions the port pins as TX or RX.
    - 1 : Pin is Transmitter
    - 0 : Pin is Receiver
  - **Status / Control Register:**
    - Status bits: Indicates the status of the handshaking bits
    - Control bits: Used to control handshaking signals and interrupt conditions

# PIA and CPU Connections




23

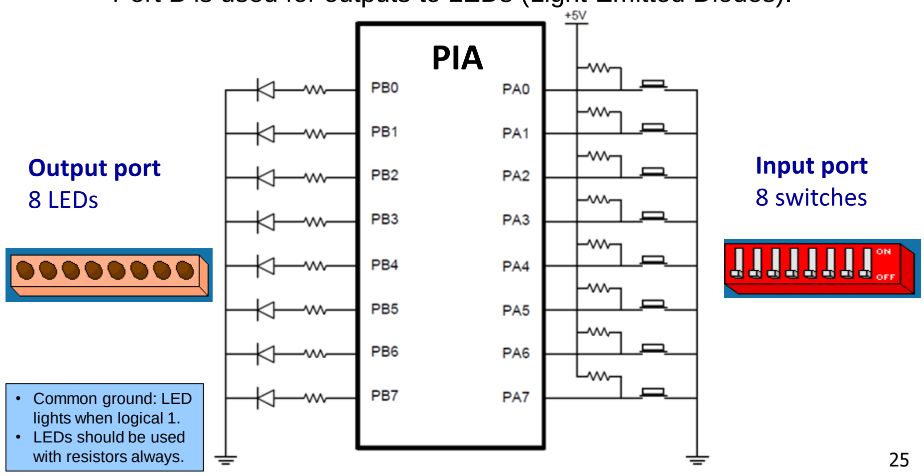
# PIA Registers




24

# EDU-PIA port connections to I/O peripherals

- The Educational PIA chip has two port registers connected to two peripheral devices permenantly.
  - Port A is used for inputs from DIP switches (ON / OFF).
  - Port B is used for outputs to LEDs (Light Emitted Diodes).






# Port names and addresses in EDU-PIA

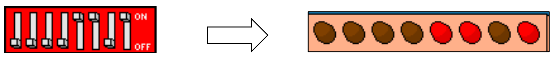
- EDU-PIA contains 6 registers.

| Register Type | Predefined Symbolic Name | Predefined Address |
|---|---|---|
| Port Registers | İSKELE.A (PORT.A) | $8080 |
| | İSKELE.B (PORT.B) | $8083 |
| Data Direction Registers | YÖNLEN.A (DIRECT.A) | $8081 |
| | YÖNLEN.B (DIRECT.B) | $8084 |
| Status/Control Registers | DURDEN.A (STATCON.A) | $8082 |
| | DURDEN.B (STATCON.B) | $8085 |

# Example1: Input /Output with PIA

- The following program runs in an endless loop.
  - Reads **switches** from PORT.A port into accumulator A.
  - Writes content of accumulator A into PORT.B port (**LEDs**).

Example : User changes some switches to ON,
then the respective LEDs are turned ON.



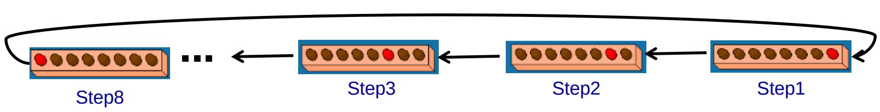

```
START
* Conditioning of PIA ports
          STA  $00,  <YÖNLEN.A>    ; All bits of Port-A are input
          STA  $FF,  <YÖNLEN.B>    ; All bits of Port-B are output

LOOP
          LDA   A,  <İSKELE.A>     ; Read from Port-A
          STA   A,  <İSKELE.B>     ; Write to Port-B
          BRA   LOOP               ; Endless loop
```

27

# Example2: Walking LED

- The following program displays a rotating LED from right-to-left.




Main program

```
START    STA $FF,  YÖNLEN.B  ;Port-B conditioned as output
         LDA A,  %00000001   ;Rightmost bit is 1 initially (Rightmost LED)

DEVAM    STA A, İSKELE.B     ;Display the LEDs
         BSR  WAIT           ;Call subroutine WAIT

         MOV  B, A           ;Copy A to B
         LSL  B              ;Shift Left B, so leftmost bit goes to Carry flag
         ROL  A              ;Rotate left A, so Carry flag goes to rightmost bit
         BRA  DEVAM          ; Endless loop
```

Wait Subroutine

```
WAIT     LDA  SK, 30000  ;Wait loop limit
AZALT    DEC  SK         ;Decrement loop counter
         BNE  AZALT      ;If not equal to zero, goto loop
         RTS             ;Return from subroutine
```

28

## Alternative1: Using STA and BSR instructions 8 times

```
START       STA $FF, YÖNLEN.B

DEVAM
        STA %00000001, İSKELE.B
        BSR  WAIT

        STA %00000010, İSKELE.B
        BSR  WAIT

        STA %00000100, İSKELE.B
        BSR  WAIT

        STA %00001000, İSKELE.B
        BSR  WAIT

        STA %00010000, İSKELE.B
        BSR  WAIT

        STA %00100000, İSKELE.B
        BSR  WAIT

        STA %01000000, İSKELE.B
        BSR  WAIT

        STA %10000000, İSKELE.B
        BSR  WAIT

        BRA DEVAM          ;Endless loop
```

29

## Alternative2: Using an array and inner loop

```
ARRAY RMB 8
  ORG ARRAY
  DAT %00000001, %00000010, %00000100, %00001000, %00010000, %00100000, %01000000, %10000000

START
      STA $FF, YÖNLEN.B
      LDA CD, ARRAY              ;Get address of array

DEVAM                            ;Inner loop
      LDA A, <CD>                ;Get data from array
      STA A, İSKELE.B             ;Write to LEDS
      BSR  WAIT

      INC D                      ;Increment inner loop counter
      CMP D, 8                   ;Compare with limit
      BLT DEVAM                  ;Goto inner loop

      BRA START                  ;Goto endless loop
```

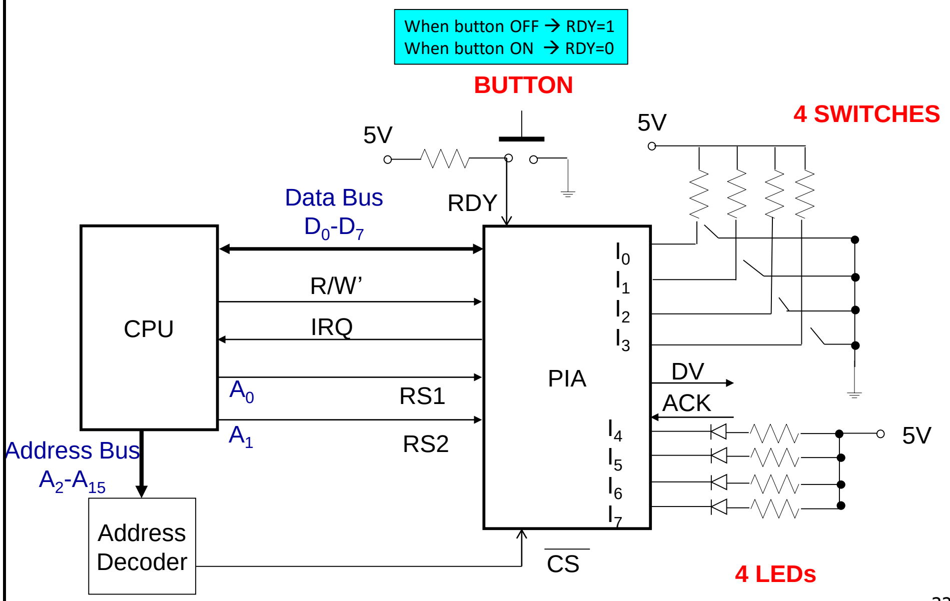
30

# Example3 :
# Using 1 Button, 4 Switches, 4 LEDs

- PIA is connected to an 8-bit microprocessor with 16-line address bus.
  - Beginning address of PIA registers is $A0A0.
  - **First four pins (0-3)** of PIA are connected to **4 switches**.
  - **Last four bits (4-7)** of PIA are connected to **4 LEDs**.
    (Type of LEDs is common voltage, which ligths when port pin is 0).
- Design a system that will illuminate the LEDs depending on the closed switches.
- The LEDs will illuminate after the user arranges the switches and then presses a **button**.
- As long as the button is kept pressed, the LEDs will continuously illuminate.
- Program should check the Status/Control register of PIA for button pressing.



# Example 3 (continued)



# Example 3 (continued)

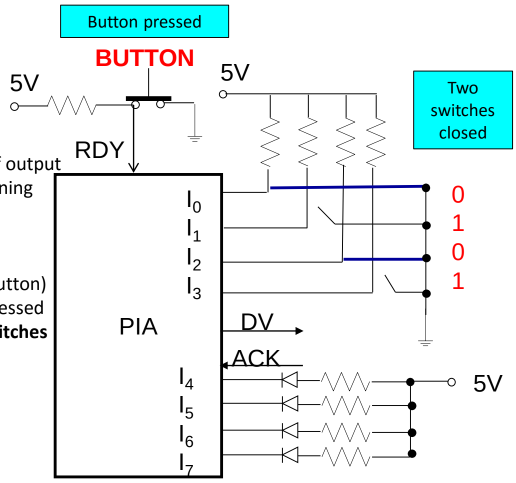
Note: Button hardware is not implemented in Mikbil.

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

START   LDA     A,$F0 ;Half input, half output
        STA     A,DIRECT   ;Conditioning
        LDA     A,$00
        STA     A,STATCON ;Reset
DEVAM   LDA     A,<STATCON>
        BIT     A,$80  ;Check RDY (button)
        BEQ     DEVAM  ;Button not pressed
        LDA     A,<PORT>   ;Read switches
```

PIA Status/Control Register :
$80 = %1000 0000

Leftmost (D7) status bit = 1 indicates
RDY = 1 (button OFF)



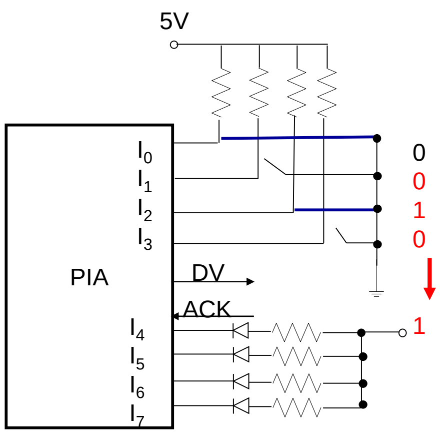

33

# Example 3 (continued)

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

START   LDA     A,$F0
        STA     A,DIRECT
        LDA     A,$00
        STA     A,STATCON
DEVAM   LDA     A,<STATCON>
        BIT     A,$80
        BEQ     DEVAM
        LDA     A,<PORT>
        LSL     A          ;I3 goes to I4
```

1st Logical shift left



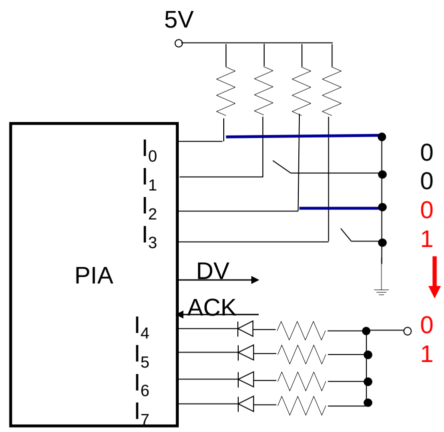

34

# Example 3 (continued)

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

START   LDA   A,$F0
        STA   A,DIRECT
        LDA   A,$00
        STA   A,STATCON
DEVAM   LDA   A,<STATCON>
        BIT   A,$80
        BEQ   DEVAM
        LDA   A,<PORT>
        LSL   A
        LSL   A
```

2nd Logical shift left



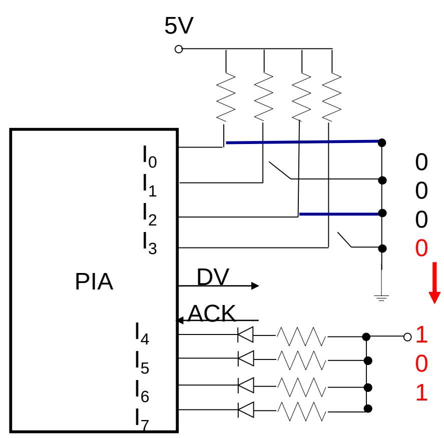

35

# Example 3 (continued)

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

START   LDA   A,$F0
        STA   A,DIRECT
        LDA   A,$00
        STA   A,STATCON
DEVAM   LDA   A,<STATCON>
        BIT   A,$80
        BEQ   DEVAM
        LDA   A,<PORT>
        LSL   A
        LSL   A
        LSL   A
```

3rd Logical shift left



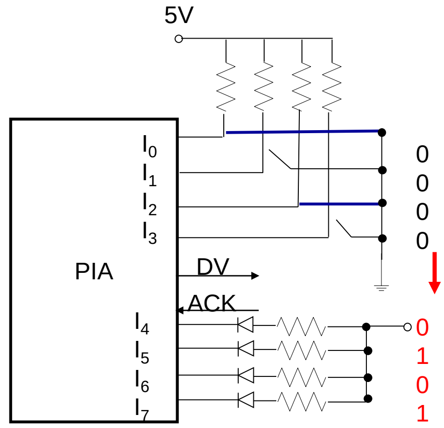

36

# Example 3 (continued)

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

START   LDA     A,$F0
        STA     A,DIRECT
        LDA     A,$00
        STA     A,STATCON
DEVAM   LDA     A,<STATCON>
        BIT     A,$80
        BEQ     DEVAM
        LDA     A,<PORT>
        LSL     A
        LSL     A
        LSL     A
        LSL     A
```

4th Logical shift left

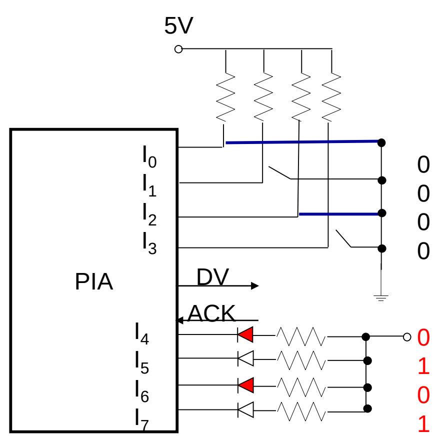





# Example 3 (end)

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

START   LDA     A,$F0
        STA     A,DIRECT
        LDA     A,$00
        STA     A,STATCON
DEVAM   LDA     A,<STATCON>
        BIT     A,$80
        BEQ     DEVAM
        LDA     A,<PORT>
        LSL     A
        LSL     A
        LSL     A
        LSL     A
        STA     A,<PORT> ;Display LEDS
        BRA     DEVAM  ;Goto loop
```




Two LEDS illuminated