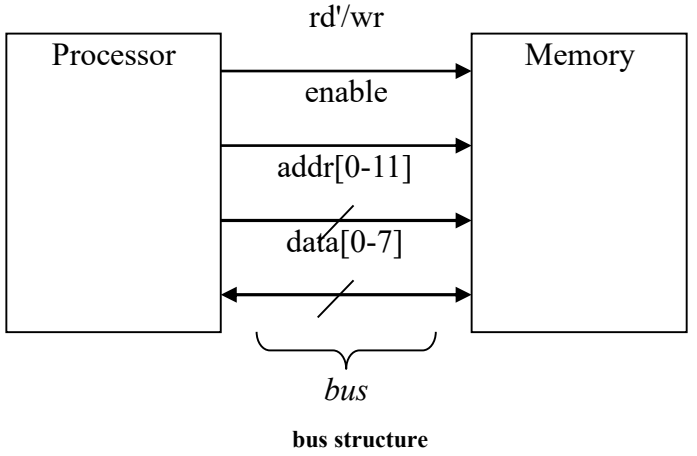
Interfacing

EE5182 Microcontrollers and Embedded Systems

Introduction

- Transfer of data among processors and memories is known as interfacing.
- Processor ⟶ Process data
- Memory ⟶ Storage
- Buses ⟶ Communication

Basic Terminology



bus structure

- Wires
  - unidirectional (rd'/wr , enable)
  - bidirectional (data)
- A set of wires with the same function :
- Bus
  - a set of wires with a single function (data bus).
  - entire wires collection along with their communication protocol.
- Protocol: rules for communicating over the wires. (low level HW protocols)

Basic Terminology

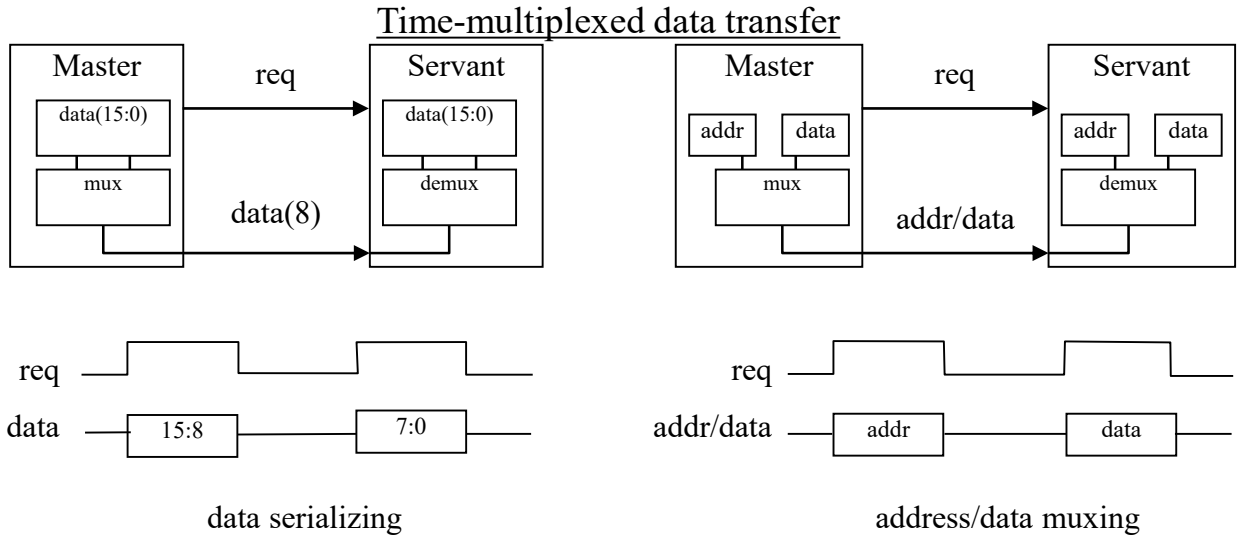
- **Port**: the actual conducting device (metal) on the processor (or memory) through which the signal is input to, or output from.
- **Pin**: a port on a processor
  - Pin is also a term referring to the extending pins from the processor (as own IC package). They are designed to be plugged into a socket on a printed-circuit board.
- If the processor coexists on a single IC with other processors and memories, pads of metal are used in the IC.

Basic protocol concepts

- **Actor**: is the processor or memory involved in data transfer.
- A protocol involves two actors: a master, and a servant (slave).
- A master initiates the data transfer (usually general-purpose processor), and the servant responds to the initiation request (usually memories and peripherals).
- **Data direction**: the direction that the transferred data moves btw. actors(receiving or sending data).
- **Addresses**: a special type of data used to indicate where regular data should go to or come from (used to address memory locations , peripherals and peripheral's registers).

Time multiplexing

- Share a single set of wires for multiple pieces of data.
- Saves wires at expense of time.



Microprocessor interfacing: I/O addressing

- The microprocessor's pins used to communicate data to and from it, are called I/O pins.
- We normally consider the access to peripherals (not memory), as I/O.
- Two common methods for using pins to support I/O : Port-based I/O (Parallel I/O), and Bus-based I/O.
- In parallel I/O , a port can be directly read and written by processor instructions, like any register.
    - Ex.P0=255, g=P2 .
    - Ports are often bit-addressable.
- In bus-based I/O, the microprocessor has a set of address, data, and control ports corresponding to bus lines, and uses the bus to access memory and peripherals.

Memory-Mapped I/O and Standard I/O

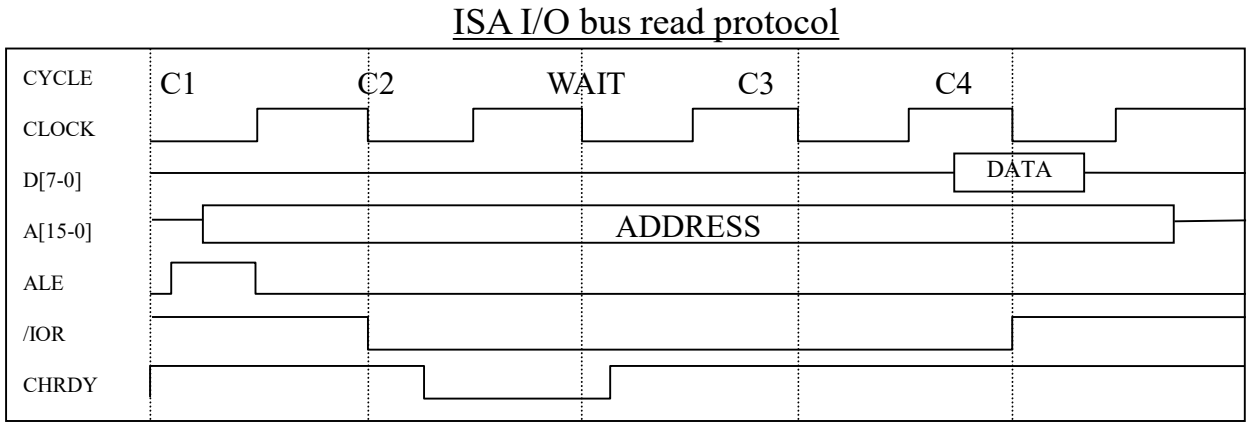
- They are two bus-based methods for microprocessor to communicate with peripherals.
- In memory-mapped I/O, peripherals occupy specific addresses in the existing address space.
    - e.g., Bus has 16-bit address, lower 32K addresses may correspond to memory, and upper 32k addresses may correspond to peripherals.
- In standard I/O (I/O-mapped I/O), the bus includes an additional pin (M`/IO), to include whether the access is to memory or peripheral.
    - e.g., Bus has 16-bit address, all of them for memory addressing if M`/IO=0, and all of them for I/O addressing if M`/IO=1.

Memory-Mapped I/O and Standard I/O

- Memory-mapped I/O
  - Requires no special instructions
    - Assembly instructions involving memory like MOV and ADD work with peripherals as well.
    - Standard I/O requires special instructions (e.g., IN, OUT) to move data between peripheral registers and memory.
- Standard I/O
  - No loss of memory addresses to peripherals.
  - Simpler address decoding logic in peripherals possible.
    - When number of peripherals much smaller than address space then high-order address bits can be ignored
      - smaller and/or faster comparators.

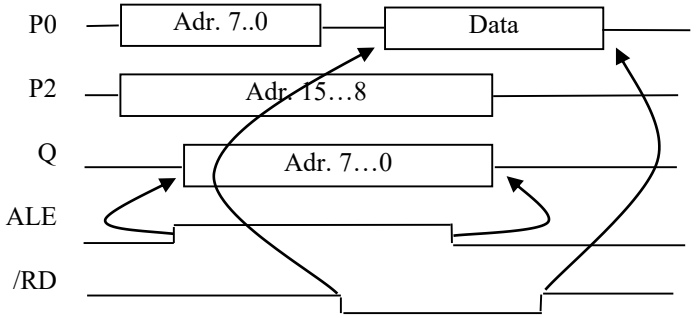
ISA bus protocol

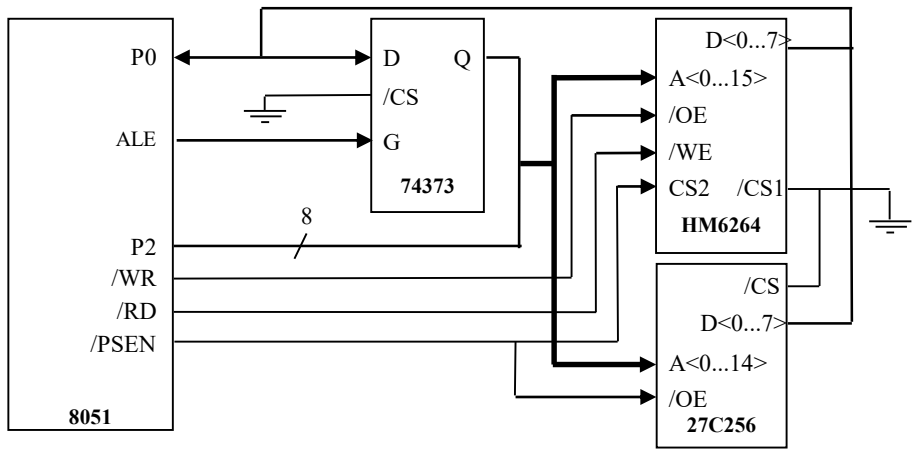
- ISA bus protocol supports standard I/O.
- The I/O address space is 16 bits, where it is 20 bits for memory.
- It uses compromise strobe/handshake control method.
- similar to memory protocol except address space.



A basic memory protocol

- Interfacing an 8051 to external memory
  - Ports P0 and P2 support port-based I/O when 8051 internal memory being used.
  - Those ports serve as data/address buses when external memory is being used.
  - 16-bit address and 8-bit data are time multiplexed; low 8-bits of address must therefore be latched with aid of ALE signal.





Interrupts (interrupt driven I/O)

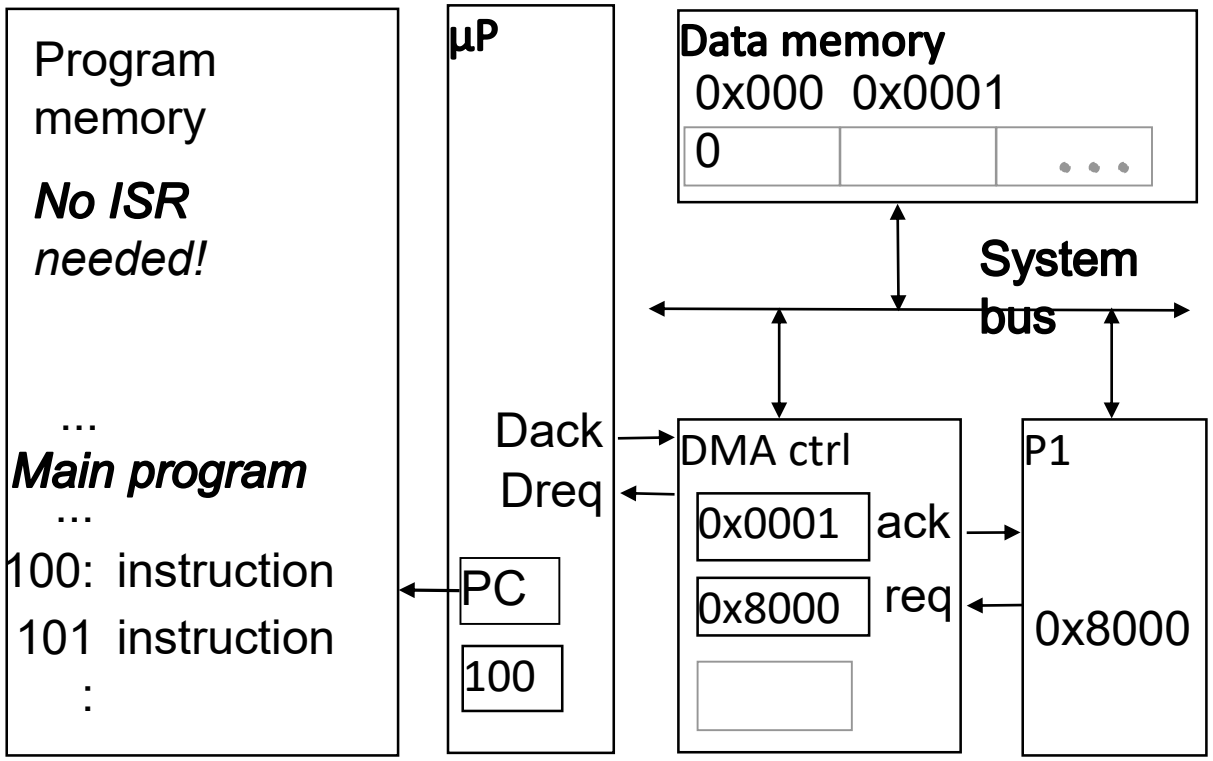
- Servicing:
  - read & process data from peripheral whenever it has new data.
  - Unpredictable
- Polling:
  - MP repeatedly check for data
  - Simple to implement
  - Waste many clock cycles
- External interrupts
  - Repeatedly MP checks Int pin after executing instruction, if asserted => jump to ISR
  - Maskable vs Non-maskable Interrupt
  - Internal Interrupt (divide by 0,…)
  - Software Interrupt .

DMA

- Buffering:
  - temporary storage of data that is awaiting processing.
- Using Interrupt:
  - Storing & restoring MP states => consuming many clock cycles (inefficient)
  - No execution during data moving.
- I/O of DMA:
  - separate single-purpose processor (DMA controller).
  - Purpose: transfer data between memories and peripherals

DMA flow of actions

1. MP is executing its main program. It has already configured the DMA ctrl registers.
2. Peripheral_1 receives input data in a register, and asserts req to request servicing by DMA ctrl.
3. DMA ctrl asserts Dreq to request control of system bus.
4. After executing instruction, MP sees Dreq asserted, releases the system bus, asserts Dack, and resumes execution. MP stalls only if it needs the system bus to continue executing.
5. DMA ctrl asserts ack reads data and (b) writes that data to memory.
6. DMA de-asserts Dreq and ack completing handshake with Peripheral_1.
7. MP de-asserts Dack and resumes control of the bus.
8. Peripheral_1 de-asserts req.

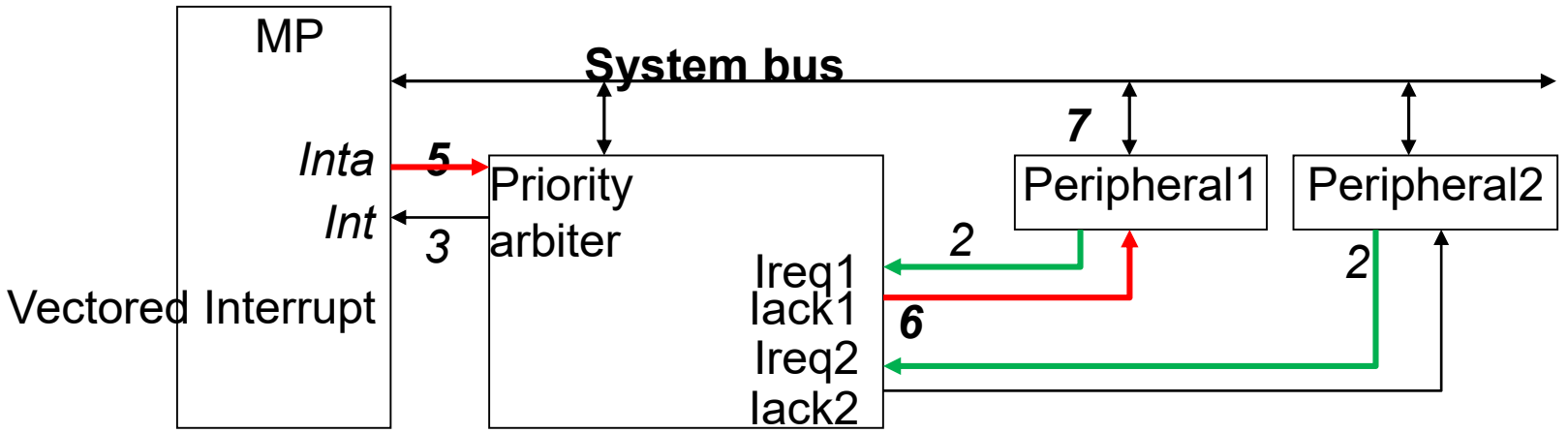


Arbitration

- Multiple peripherals request service simultaneously from single MP or single DMA
- Arbitration: decide which one get services.
  - Priority Arbiter.
  - Daisy-Chain Arbitration.
  - Network-Oriented Arbitration Methods.

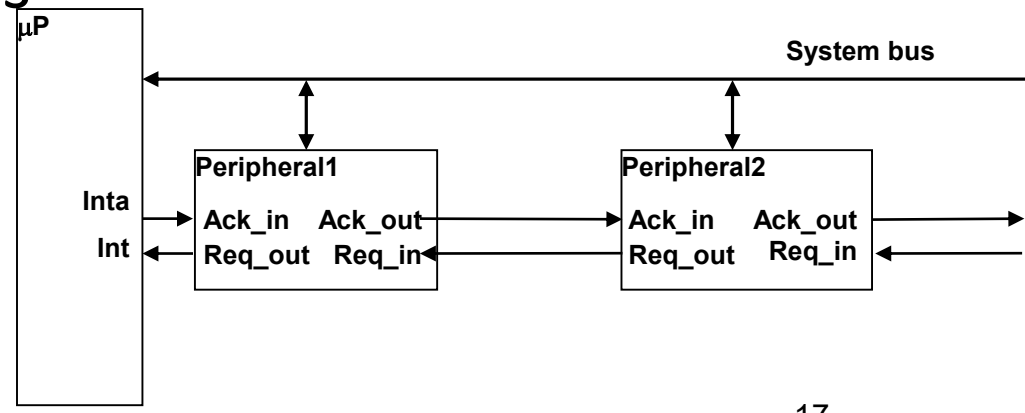
Priority Arbiter

- Is a single-purpose processor
- 2 schemes to determine priority among peripherals:
  - Fixed priority: unique rank for each peripheral. Arbiter choose the higher rank.
  - Rotating priority (round-robin): based on history of servicing (e.g. least recently serviced)
    - More equitable of servicing.



Daisy-Chain Arbitration

- Peripherals connected as a chain
  - Each peripheral has: req. output, ack. input, req. input, and ack. Output
- Add or remove peripherals without redesigning the system
- Peripherals at the end of chain could become intolerably slow.
- Isn't supporting more advanced priority schemes
- If one peripheral stop, the other lose access
- Each peripheral must be daisy-chain aware
  - Otherwise, external logic is used.



Network-Oriented Arbitration

- Multiple MP connected by a shared bus (network).
- Arbitration among processors.
  - Typically built right into the bus protocol
- The protocol must ensure that no contending processors sending at the same time
  - Examples: I2C, Ethernet, CAN …

Multilevel Bus Architectures

- Numerous type of communications:
  - Most frequent and high speed (between MPs).
  - Less frequent and low speed (between MP and Peripherals like UART)
- Single high speed bus:
  - Required each peripheral to have high-speed bus interface
    - Extra gates ,Power consumption and cost.
  - May not be very portable.
  - May result in slower bus.
- 2 level buses:
  - Processor local bus: connects MP, cache, memory controllers …
  - Peripheral bus: ISA, PCI …
    - emphasize portability, low power or low gate count.
  - Bridge (single-purpose processor) connect two bus levels

Multilevel Bus Architectures cont.

- 2 level buses: VSI Alliance.
  - Processor local bus
  - System bus
  - Peripheral bus