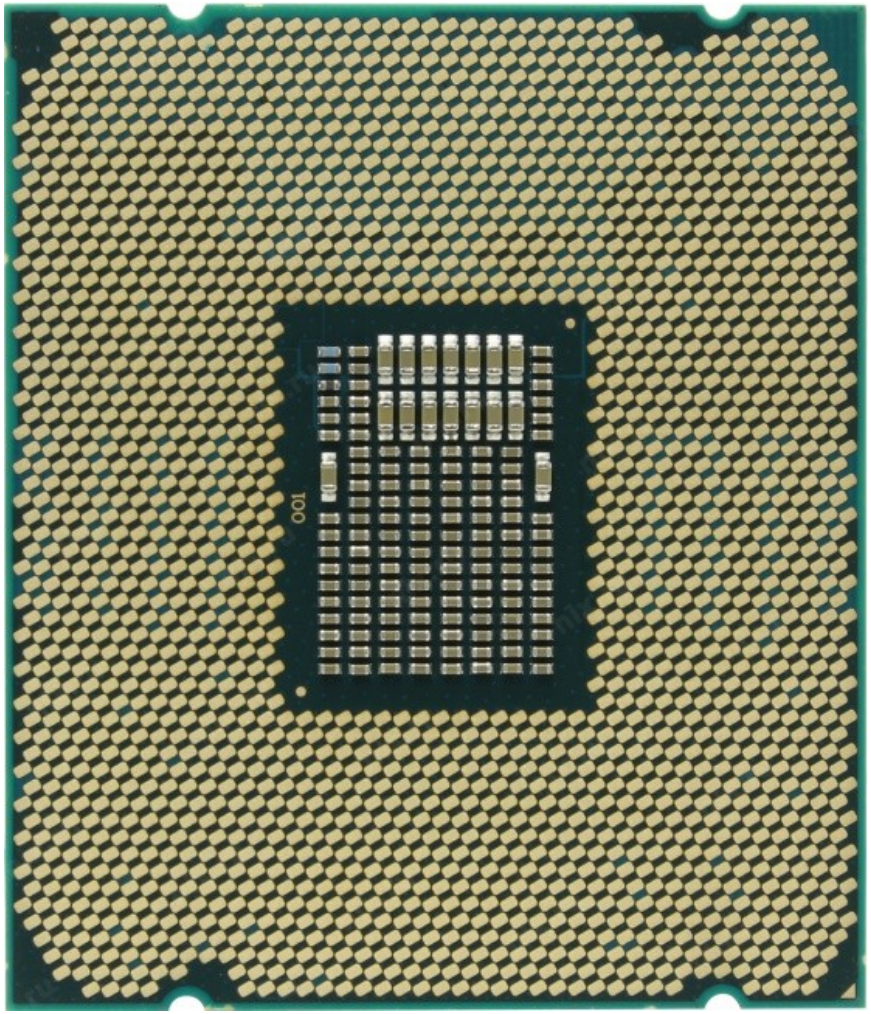
Ввод-вывод. Шины данных
Ядро операционной системы

Хеллоуинская лекция по АКОС

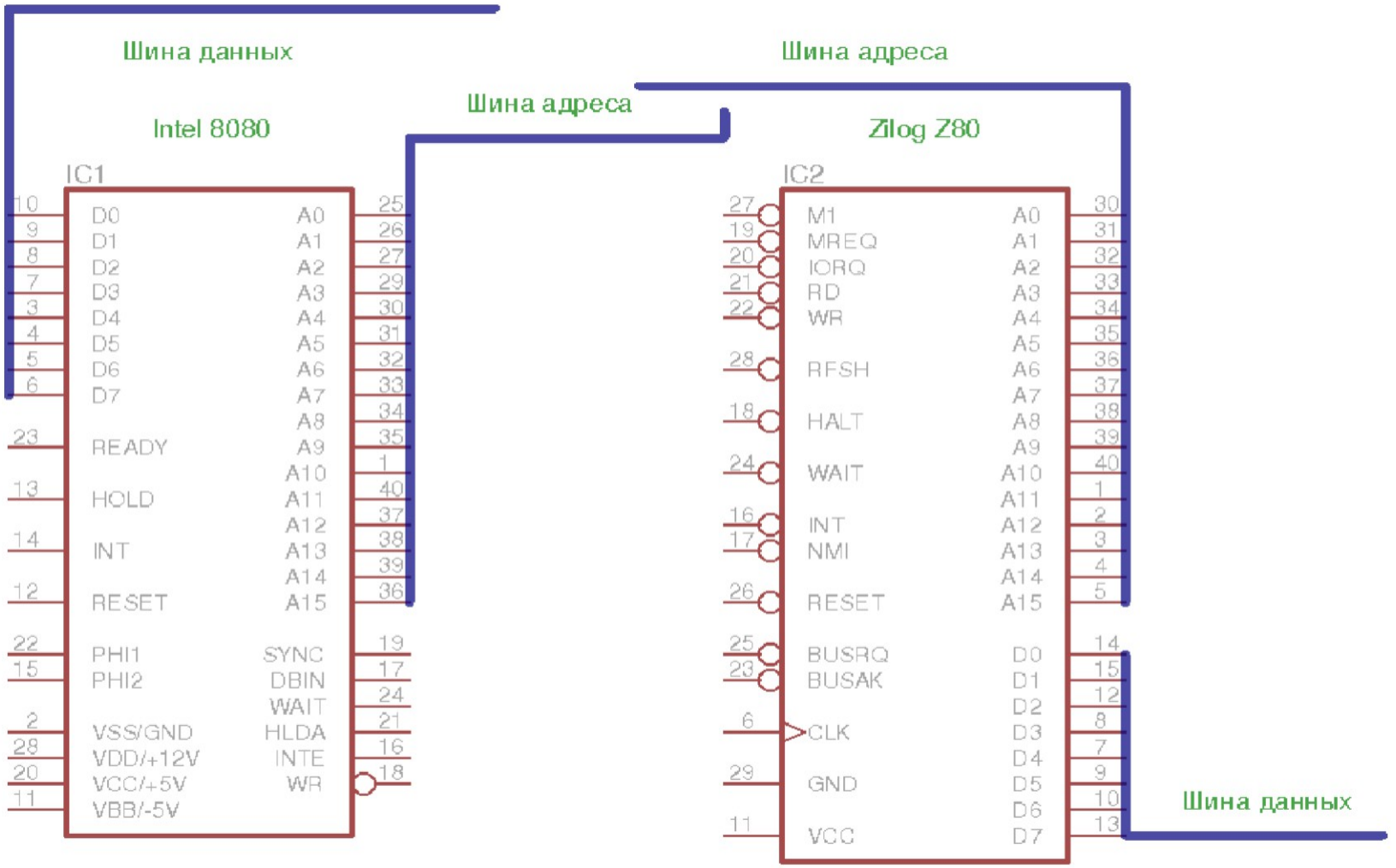
Core i9, вид снизу



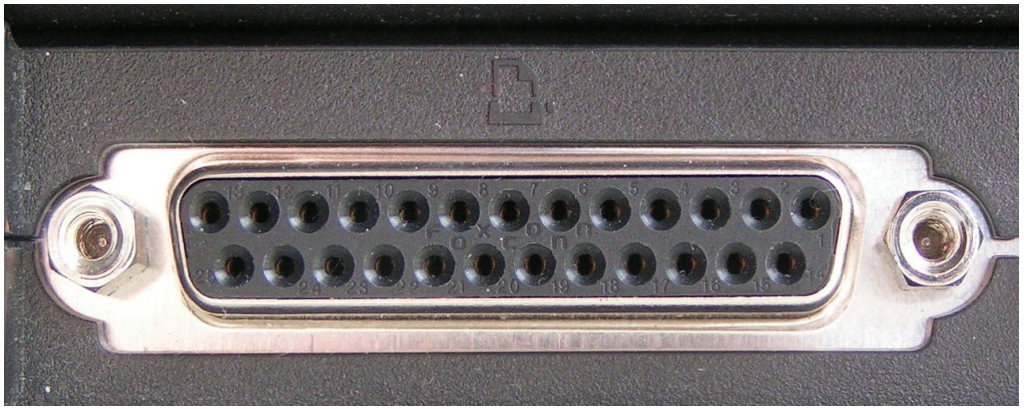
Виды шин

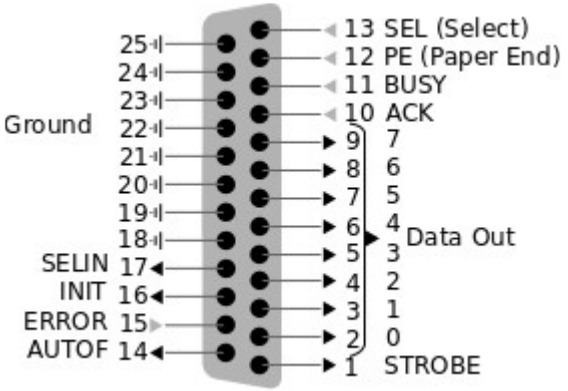
- Последовательные:
  - I2C / SPI
  - UART
  - USB
  - SATA
  - Ethernet
  - PCI Express
- Параллельные:
  - Шина адреса/данных процессора
  - GPIO / LPT / IDE
  - ISA / PCI / AGP

Параллельные шины



Параллельный LPT порт

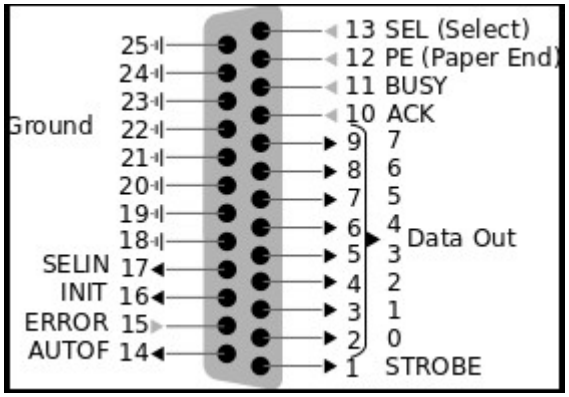




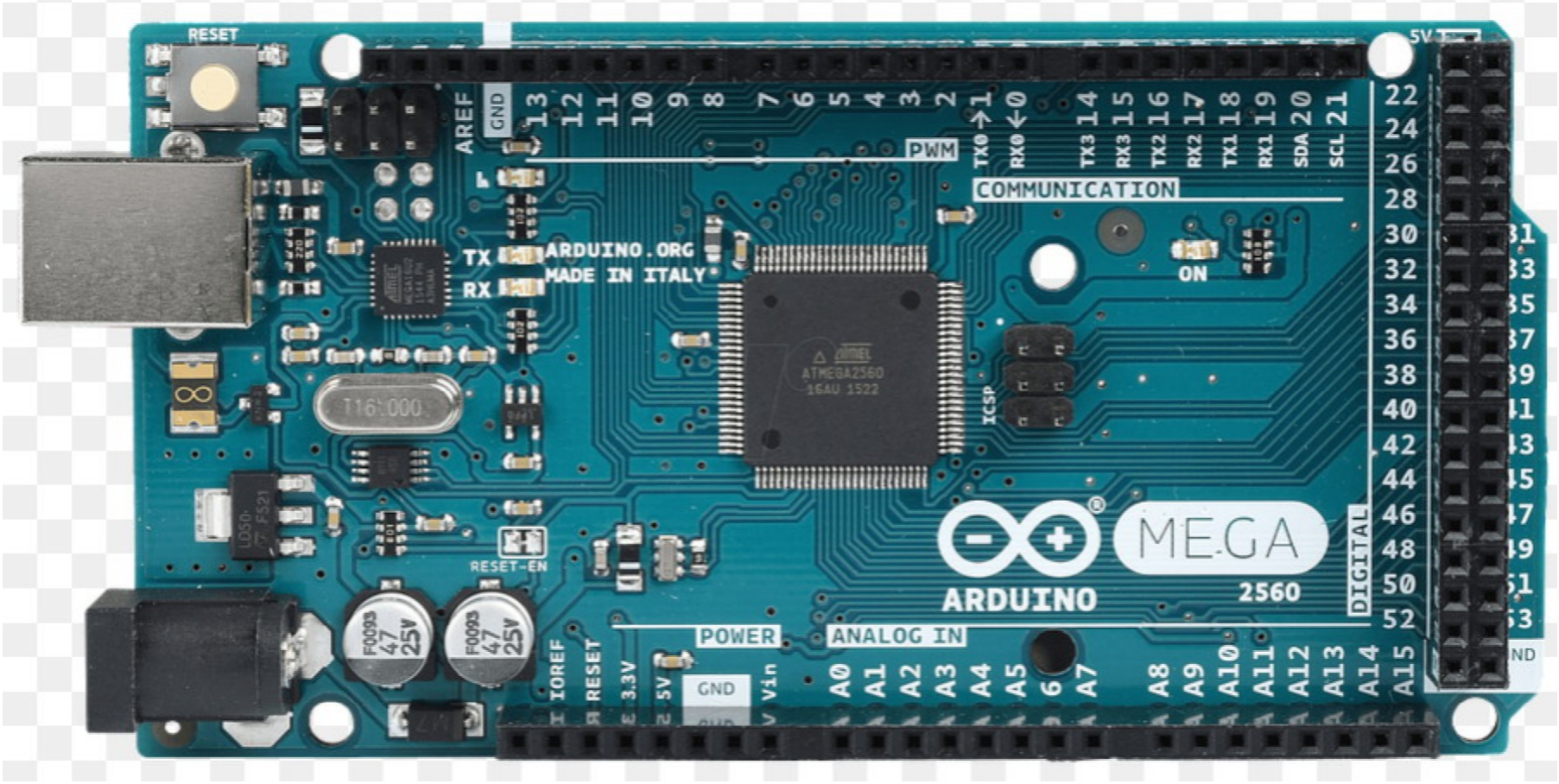
Параллельные шины

- Каждый контакт - один бит
- Количество контактов обычно кратно 8
- Данные передаются размерами, кратными 1 байту
- Отделные контакты можно использовать независимо - с побитовой маской

Однокнопочный интерфейс

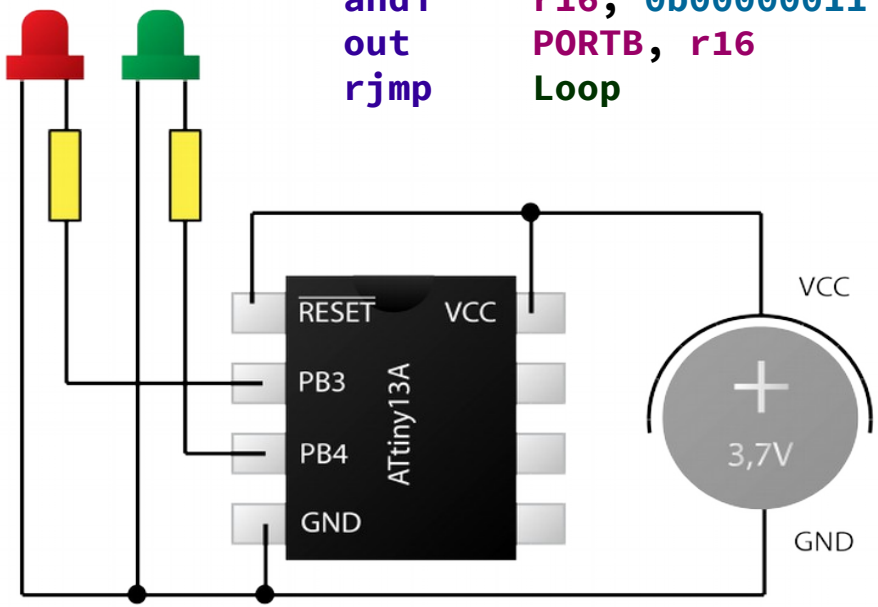


General Purpose In/Out (GPIO)

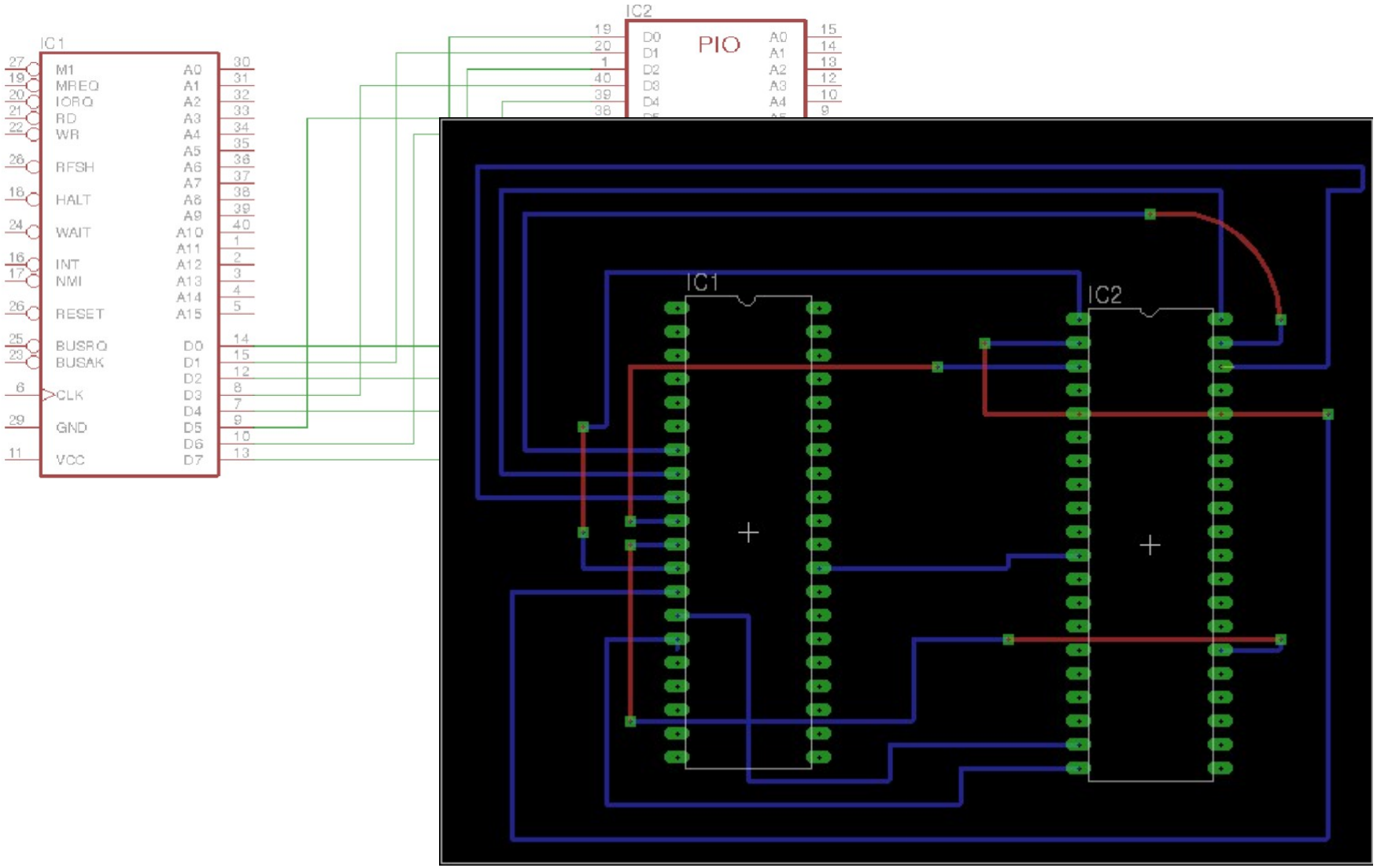


Использование GPIO

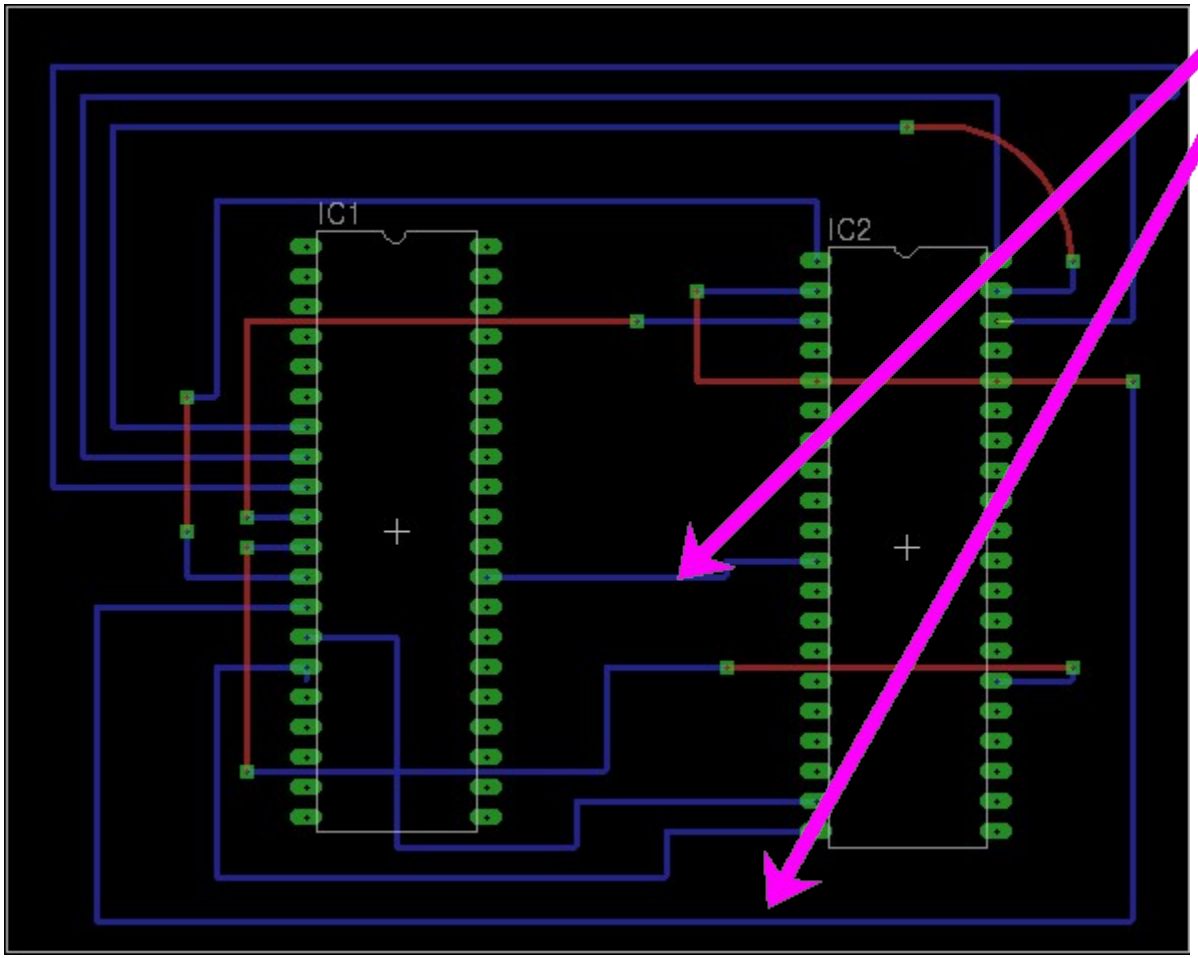
```
        .include "tn45def.inc"
        .cseg
        ldi     r16, 0b00000011
        out     DDRB, r16           ; set pins PB0, PB1 to out
        ldi     r16, 0b00000001
        out     PORTB, r16          ; set pin PB0 to '1'
Loop:   in      r16, PORTB          ; read value from PBx
        com     r16                 ; invert bits
        andi    r16, 0b00000011     ; apply mask for PB0 and PB1
        out     PORTB, r16          ; write value back to PBx
        rjmp    Loop                ; continue
```



Параллельная шина на плате



Параллельная шина на плате



D0: Длина 3 см

D7: Длина 25 см

Скорость распространения сигнала ~60% от скорости света **(грубая верхняя оценка, без учета волнового сопротивления)**:

V=1.8*10**7 м/с

S0 = 0.03 м

S7 = 0.25 м

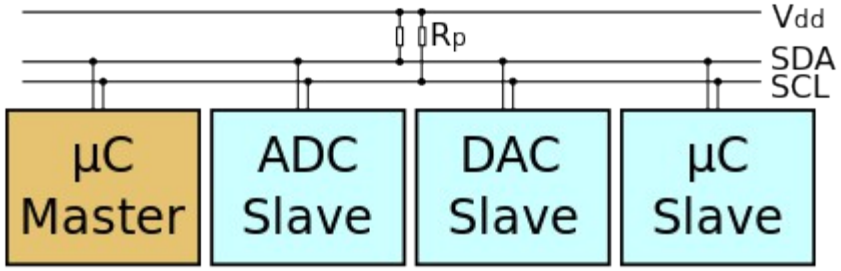
Freq0 = 6000 МГц

Freq7 = 720 МГц

Последовательные шины

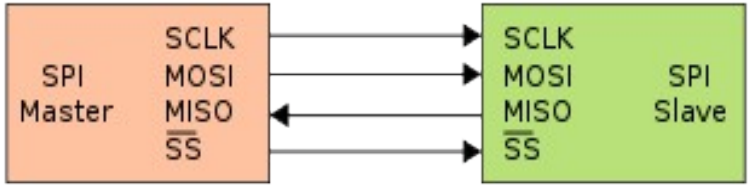
- Данные передаются последовательно
- Обычно разделяют направление (TX/RX)
- Могут работать на высоких частотах
- Требуется аппаратная буфферизация для того, чтобы иметь доступ к данным
- На больших расстояниях используется физическая дифференциальная пара

I2C и SPI



I2C:

- два провода + GND
- широковещательная передача информации

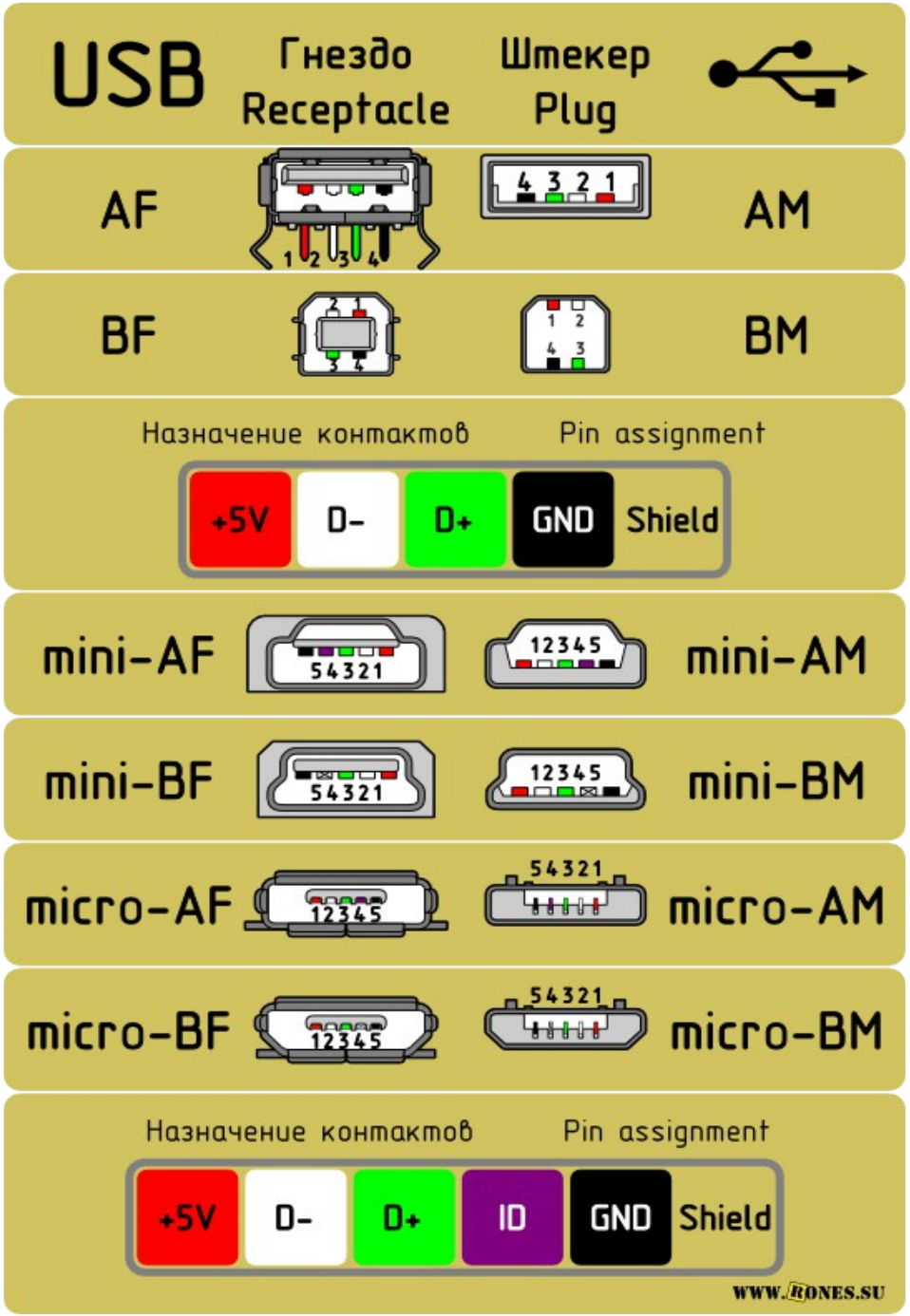


SPI:

- независимый двусторонний канал передачи данных
- отдельный сигнал Slave Select

UART

- Двунаправленная передача данных
- Простая реализация буфера
- Стандарт определяет только логический, но не физический уровень
- Примеры реализации: Arduino/Raspberry Pi (0...5V), RS-232 (-15...15V), RS-432 (-12...12V, дифпара)



USB 2.x

- +5V - питание
- GND - питание
- D+ - положительный сигнал дифпары
- D- - инвертированный сигнал дифпары
- ID - выбор Master/Slave:
  - Если ID соединен с GND - Master, если не соединен - Slave

PCI / AGP

- Параллельная 32 или 64 битная шина
- Мультиплексированная передача A/D
- Частоты до 66.6МГц
- Максимальная скорость 533Мбайт/с
- Данные передаются в виде "сообщений"
- **Все устройства используют одну шину**
- Отдельное устройство - арбитр шины
- Могут иметь прямой доступ к памяти

- Шина AGP - отличается от PCI выделенным доступом к процессору/памяти

PCI Express

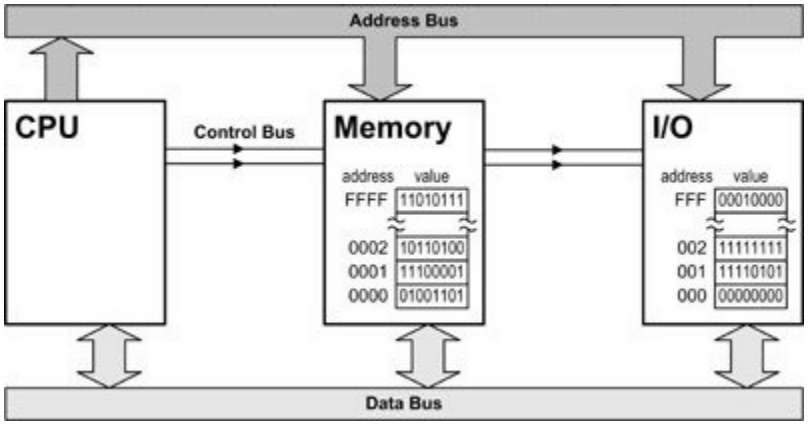
- Последовательная передача данных, используется дифференциальная пара
- Одноваременно может использовать много пар (x1, x2, x4, x8, x12, x16, x32)
- Скорость - 250М/с на каждую пару
- Вместо общей шины используется коммутатор устройств, подключенный к процессору

Способы взаимодействия с устройствами

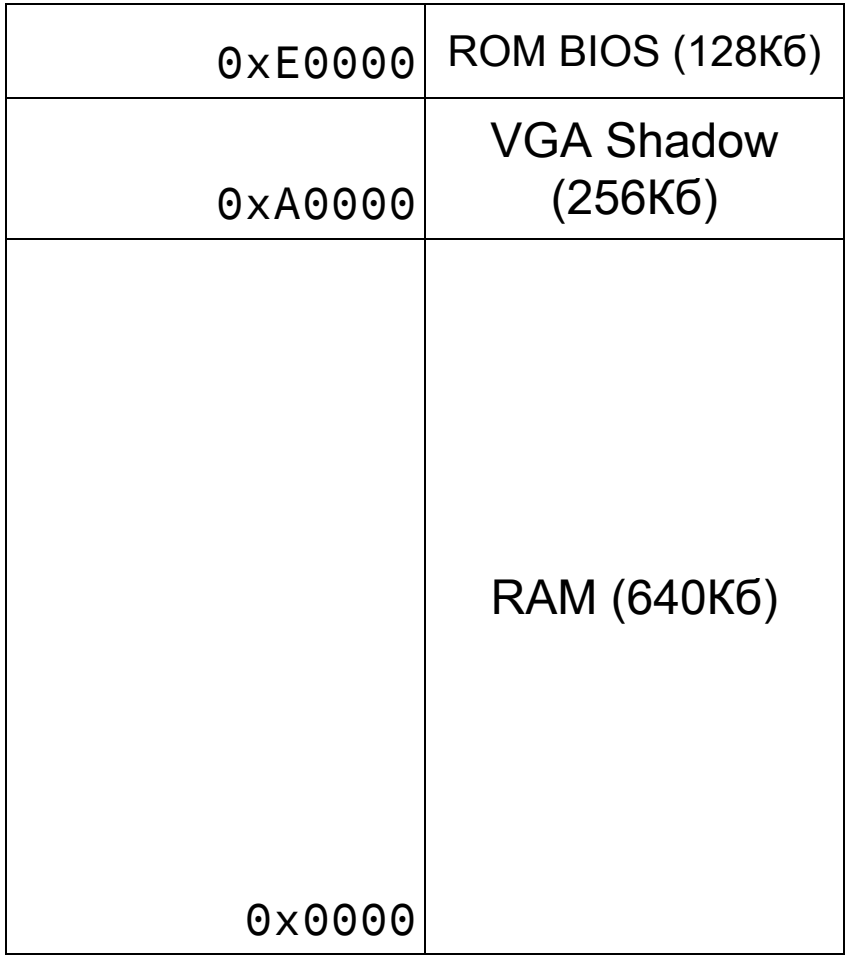
- Через зарезервированные адреса в адресном пространстве:
  - VGA Shadow на x86
  - ARM-процессоры
- Через порты ввода/вывода - ортогонально адресному пространству:
  - x86
  - AVR
- Без участия процессора - Direct Memory Access для устройств

Порты ввода-вывода на x86

- Номер порта - 16-битное слово
- За каждым устройством закрепляется фиксированный номер порта на этапе P'n'P или выставлением перемычек
- Команды **`in`**, **`out`** (и их варианты с разными суффиксами) взаимодействуют с портами
- **Ввод-вывод через порты - затратная операция**



Физическая память в Real Mode



- После включения процессор находится в 16-битном реальном режиме
- Доступен только 1Мб адресного пространства
- Память делится на сегменты по 64К

Стадии включения (PC)

1. Выполенение программы из BIOS, инициализация устройств
2. Чтение MBR (512 байт) с загрузочного диска
3. Выполнение кода из MBR. На этом этапе процессор может переключиться в защищенный режим
4. Выполнение ядра в реальном (DOS) или защищенном (*nix/Windows) режиме

Стадии включения (EFI/UEFI)

1. Выполенение программы из BIOS, инициализация устройств
2. ~~Чтение MBR (512 байт) с загрузочного диска~~
3. ~~Выполнение кода из MBR. На этом этапе процессор может переключиться в защищенный режим~~
4. Выполнение ядра из EFI-NVRAM в защищенном режиме

Ядро

- Обычная программа ELF или M$ PE, хранится где-то на диске, возможно в подкаталоге ФС
- Запускается до того, как процессор понизил привилегии:
  - доступ ко всей памяти
  - доступ к портам ввода-вывода

Задачи ядра ОС x86

- Переинициализировать шины PCI/USB
  мы не доверяем BIOS'у после перехода в защищенный режим
- Найти и загрузить все *драйверы* устройств в соответствии с PCI ID.
- Инициализировать новый вектор прерываний
- Понизить уровень привилегий процессора и запустить процесс `init`

Взаимодействие с ядром

- Пользовательские программы не имеют доступа к портам I/O и физической памяти
- Память каждой программы уникальна. Таблицу отображения виртуальной памяти на физическую настраивает ядро для каждого процесса
- Выйти за пределы ограничений процесс может только обратившись к ядру:

```
int 0x80
```