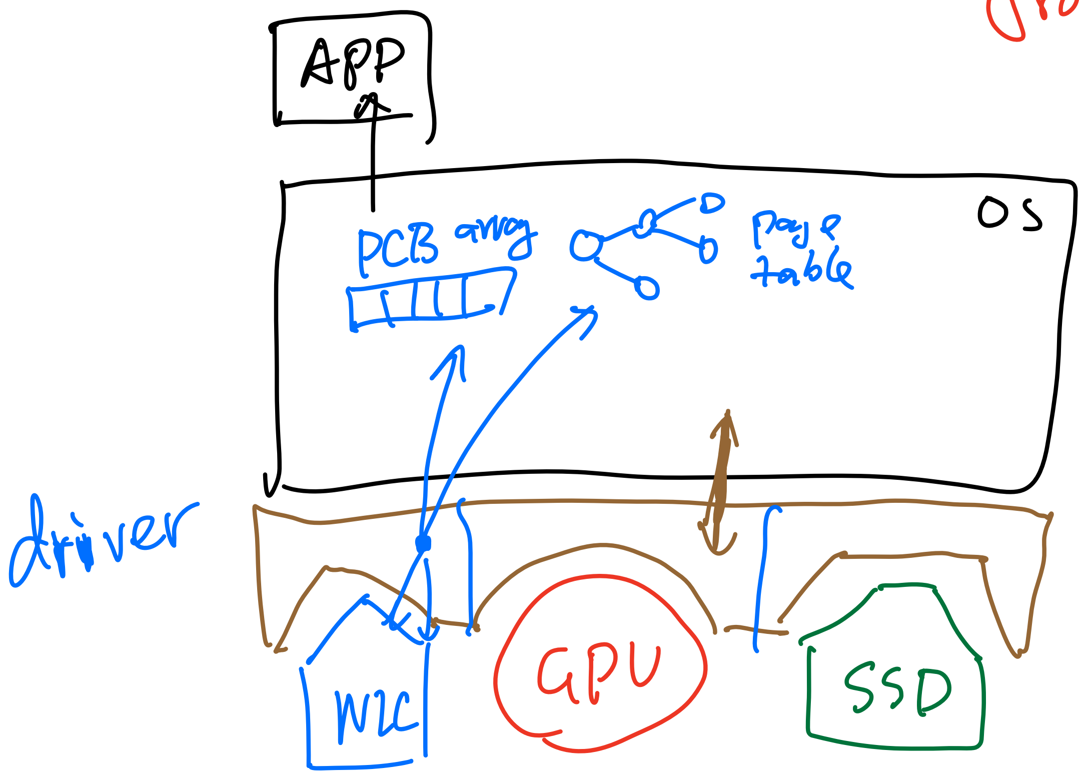
week 10
cs4973/cs6640
03/10 2025
https://naizhengtan.github.io/25spring/

☑ admin: exam and final project
☑ 1. Device drivers
☑ 2. Mechanics of communication
□ 3. An example: a tty dev :C
□ 4. Communication configurations
☑ 5. Hints about Lab6 (SD card driver)

- exam : week12
- final project :
  - week13's Monday
  - safe vs. risky

· Device driver

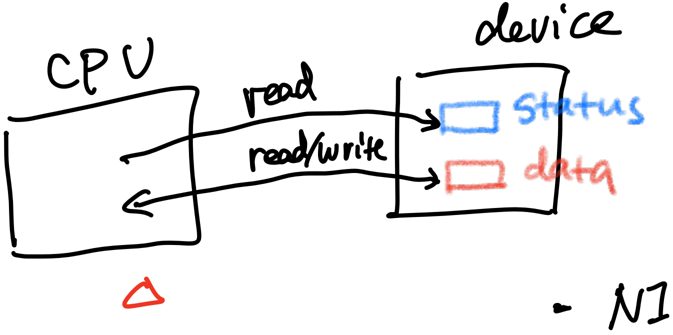


Q: GPU drive from NVIDIA for AMD GPUs

NO

Q: NVIDIA 4090, driver for WIN work for Linux?

No

- 

Q: interface between CPU↔device?

- NIC:
  - addr (4B), KBs buffer
  - "fetch"

  CPU
- I2C, game Controller
- SSD

  interrupt
- USB, flash drive
  - special addr
  - store/load (standard)

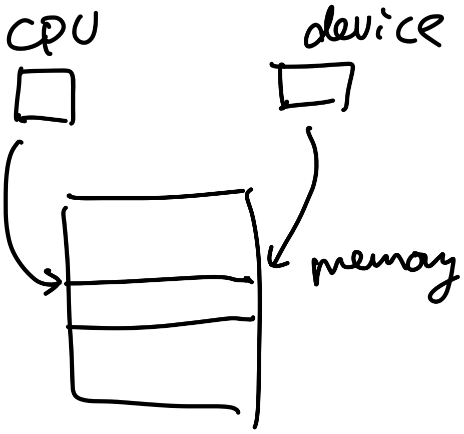
- Mechanics of communication between CPU and devices

(a) explicit I/O instructions ←

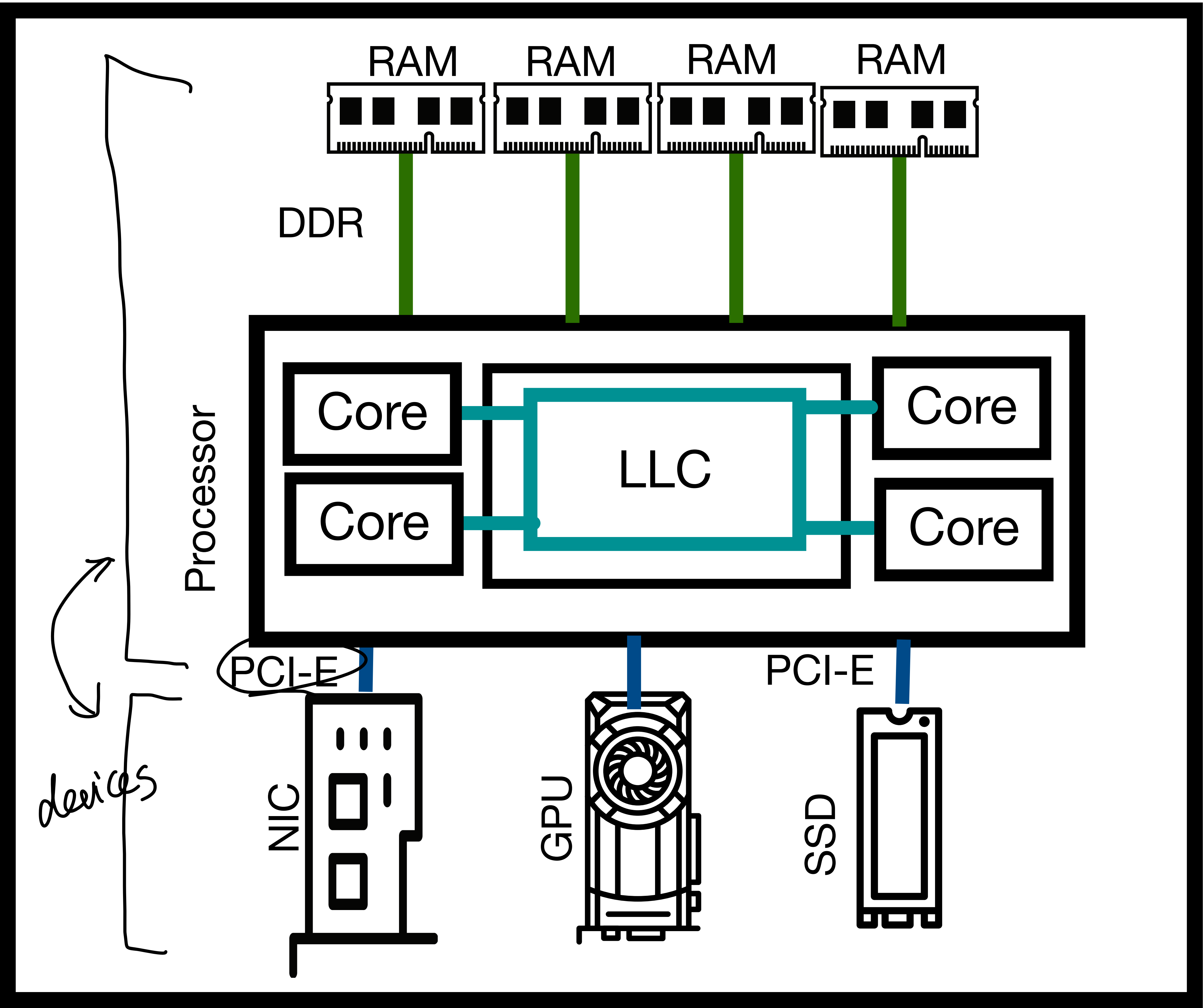
(b) memory-mapped I/O ←

(c) interrupts

(d) through memory

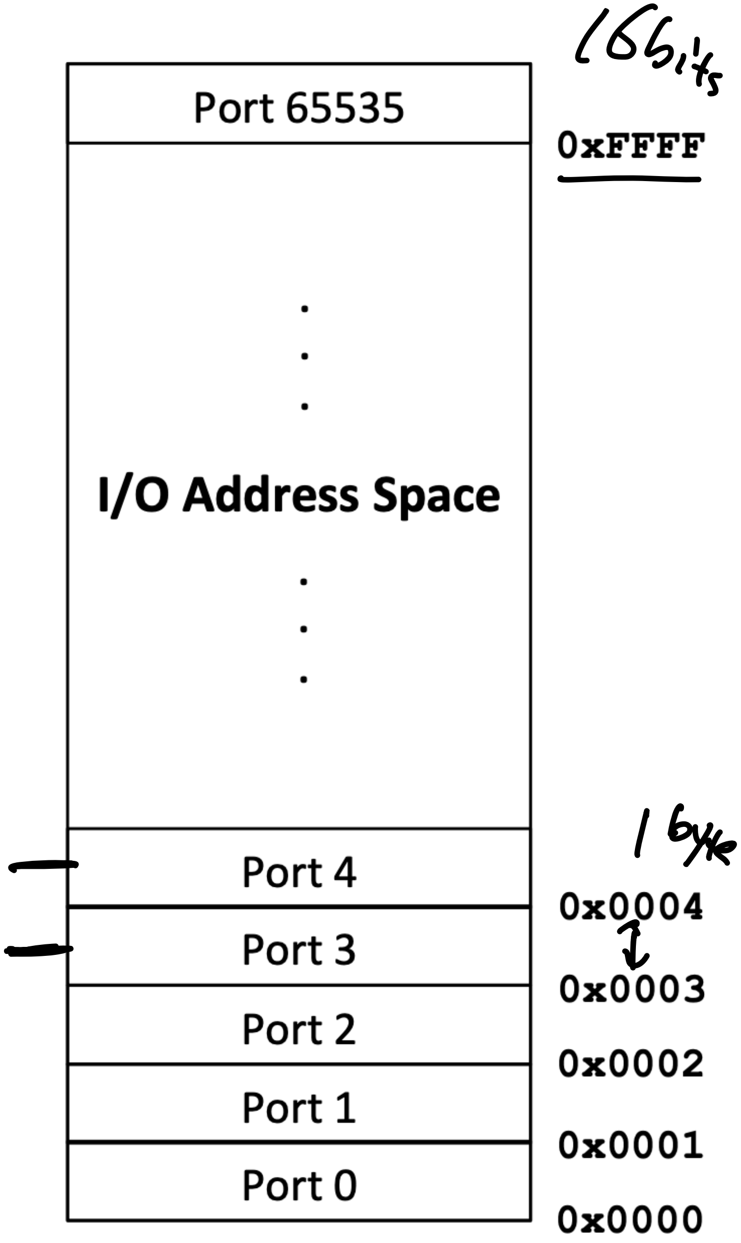


Machine


RAM
RAM
RAM
RAM
DDR
Processor
Core
Core
LLC
Core
Core
PCI-E
PCI-E
NIC
GPU
SSD
devices

Port I/O Address Space

- Software and hardware architectures of x86 architecture support a separate address space called "I/O Address Space"
  - Separate from memory space
- Access to this separate I/O space is handled through a set of I/O instructions
  - IN,OUT, INS, OUTS
- Access requires Ring0 privileges
  - Access requirement does not apply to all operating modes (like Real-Mode)
- The processor allows 64 KB+3 bytes to be addressed within the I/O space
- Harkens back to a time when memory was not so plentiful
- You may never see port I/O when analyzing high-level applications, but in systems programming (and especially BIOS) you will see lots of port I/O
- One of the biggest impediments to understanding what's going on in a BIOS



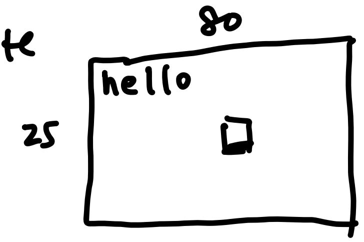
Intel Programmer's guide, Vol 1, 16.1

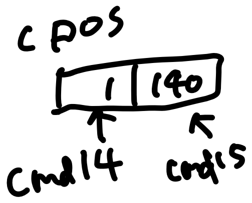
```
OSI Week10: I/O and device driver

1. An example of I/O instructions:
   Setting the cursor position

  The code below is excerpted from WeensyOS's k-hardware.c. It
  uses I/O instructions to set a blinking cursor in the upper left of
  the screen.

  // console_show_cursor(cpos)                80x25
  //   Move the console cursor to position ‘cpos‘,
  //   which should be between 0 and 80 * 25.           0x3D4

  void console_show_cursor(int cpos) {                   0x3D5
    if (cpos < 0 || cpos > CONSOLE_ROWS * CONSOLE_COLUMNS)
      cpos = 0;

    outb(0x3D4, 14); // Command 14 = upper byte of position
    outb(0x3D5, cpos / 256); // ~~row~~ data
    outb(0x3D4, 15); // Command 15 = lower byte of position
    outb(0x3D5, cpos % 256); // ~~column~~

  }
```

instruction

addr

1byte





2^{16}

```
2. Memory-mapped I/O

  a. Here is a 32-bit PC's physical memory map:              x86

      +------------------+ <- 0xFFFFFFFF (4GB)
      |      32-bit      |
      |  memory mapped   |
      |     devices      |
      |                  |
      /\/\/\/\/\/\/\/\/\/\

      /\/\/\/\/\/\/\/\/\/\
      |                  |
      |      Unused      |
      |                  |
      +------------------+ <- depends on amount of RAM
      |                  |
      |                  |
      | Extended Memory  |
      |                  |
      |                  |
      +------------------+ <- 0x00100000 (1MB)
      |     BIOS ROM     |
      +------------------+ <- 0x000F0000 (960KB)
      |  16-bit devices, |
      |  expansion ROMs  |
      +------------------+ <- 0x000C0000 (768KB)
      |   VGA Display    |
      +------------------+ <- 0x000A0000 (640KB)
      |                  |
      |    Low Memory    |
      |                  |
      +------------------+ <- 0x00000000

  [Credit to Frans Kaashoek, Robert Morris, and
  Nickolai Zeldovich for this picture]
```

b. Loads and stores to the device memory "go to hardware".

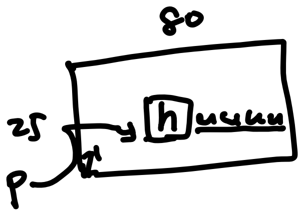
An example is in the console printing code from WeensyOS. Here is an excerpt from link/shared.ld:

```
/* Compare the address below to the map above. */
PROVIDE(console = 0xB8000);
```

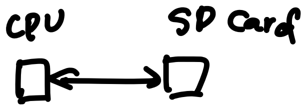
This is an excerpt from lib.c; notice how it uses the address "console":

```
/*
* prints a character to the console at the specified
* cursor position in the specified color.
* Question: what is going on in the check
* if (c == '\n')
* ?
* Hint: '\n' is "C" for "newline" (the user pressed enter).
*/
static void console_putc(printer* p, unsigned char c, int color) {
  console_printer* cp = (console_printer*) p;
  if (cp->cursor >= console + CONSOLE_ROWS * CONSOLE_COLUMNS) {
    cp->cursor = console;
  }
  if (c == '\n') {
    int pos = (cp->cursor - console) % 80;
    for (; pos != 80; pos++) {
      *cp->cursor++ = ' ' | color;
    }
  } else {
    *cp->cursor++ = c | color;
  }
}
```

='h'

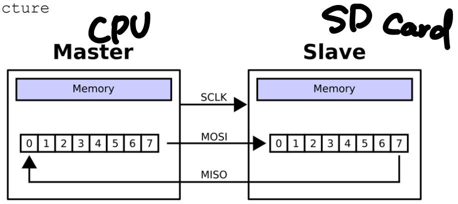


Write "c" to the position

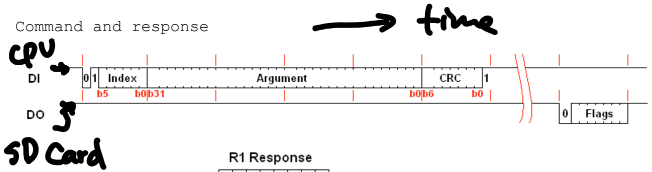


3. SPI basics

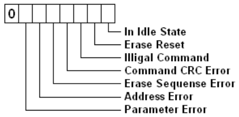
- SPI architecture



- Command and response

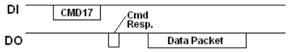


R1 Response

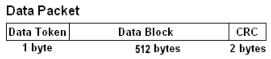


512Bytes

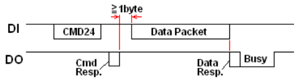
- Single-block read



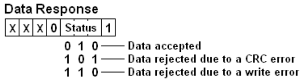
- Data packet



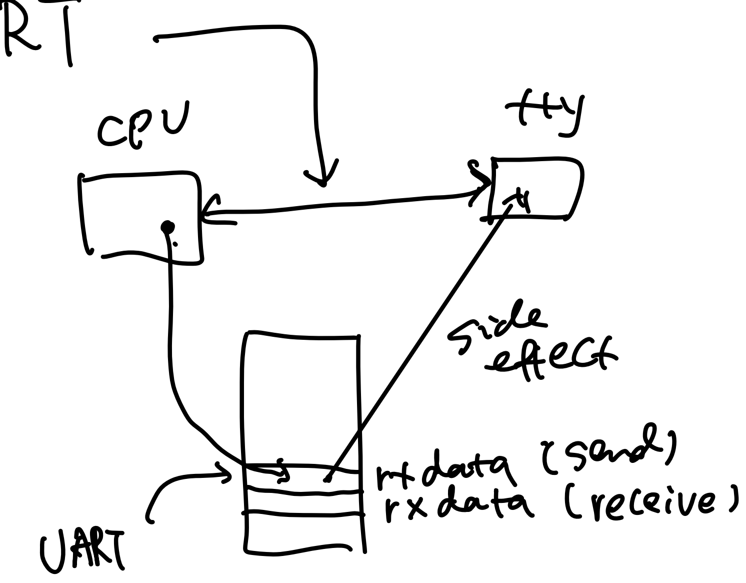
- Single-block write



- Data response

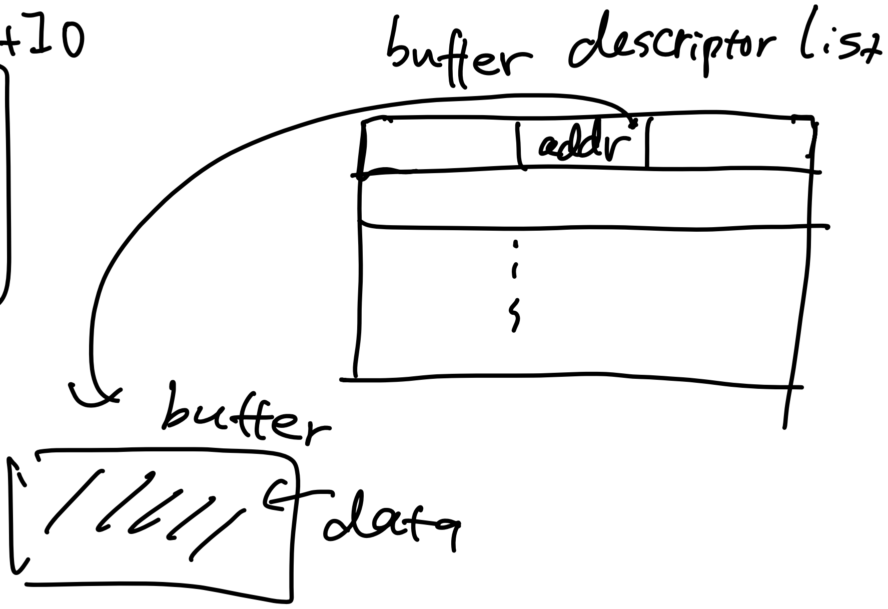

Data Response
X X X 0 Status 1
0 1 0 Data accepted
1 0 1 Data rejected due to a CRC error
1 1 0 Data rejected due to a write error

- UART



- Communication configurations
  - status : polling vs. interrupt
  - data : Programmed I/O vs. DMA
    - MMIO
    - PortIO

$$\{\text{polling}, \text{interrupt}\} \times \{\text{PIO}, \text{DMA}\}$$


buffer descriptor list
addr
buffer
data