# 操作系统研讨课

蒋德钧

Fall Term 2019-2020

email: jiangdejun@ict.ac.cn
office phone: 62601007

University of Chinese Academy of Sciences

# Lecture 5 Device Driver

2019.12.02

University of Chinese Academy of Sciences

# Schedule

- Project 5 assignment
- Project 4 due
  - 如果课程网站无法提交，请将作业打成压缩包提交至如下邮箱
  - ucasos2019@163.com
  - 邮件和压缩包都命名为：学号-姓名-P4

University of Chinese Academy of Sciences

# Project 5 Device Driver

- Requirement
  - Implement driver for network card
    - Setup MAC controller registers to allow network card initialization, and sending/receiving data without interrupt
    - Implement sending/receiving data with blocking
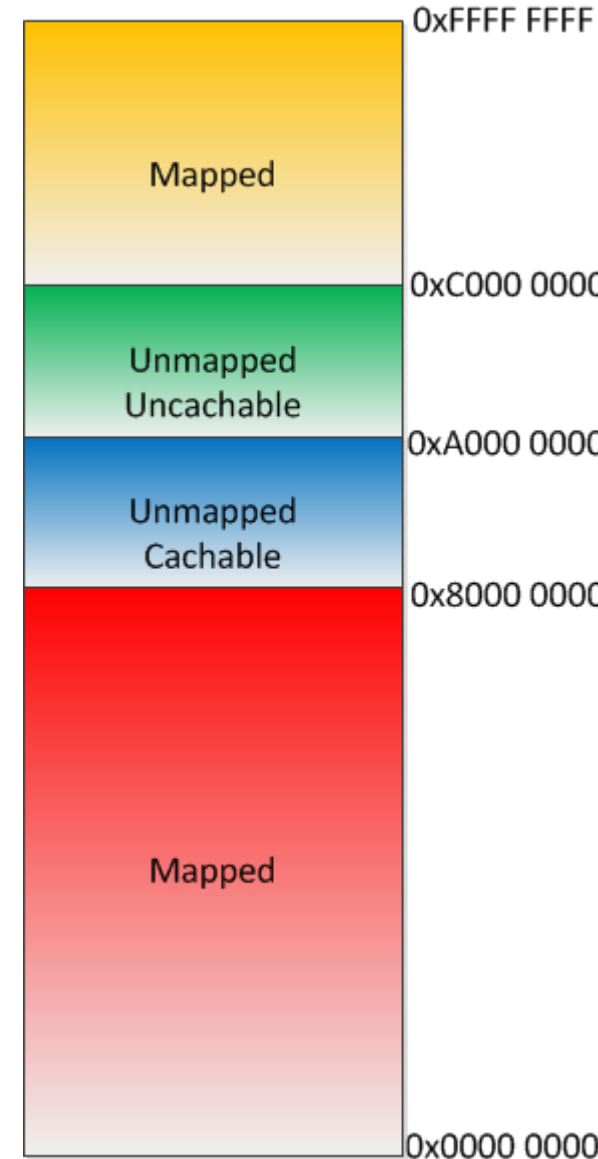    - Implement network interrupt handler to serve receiving data

# Project 5 Device Driver

- Device driver
  - Interact with device controller
  - Issue commands to device controller to finish data reads and writes, e.g. network and disk
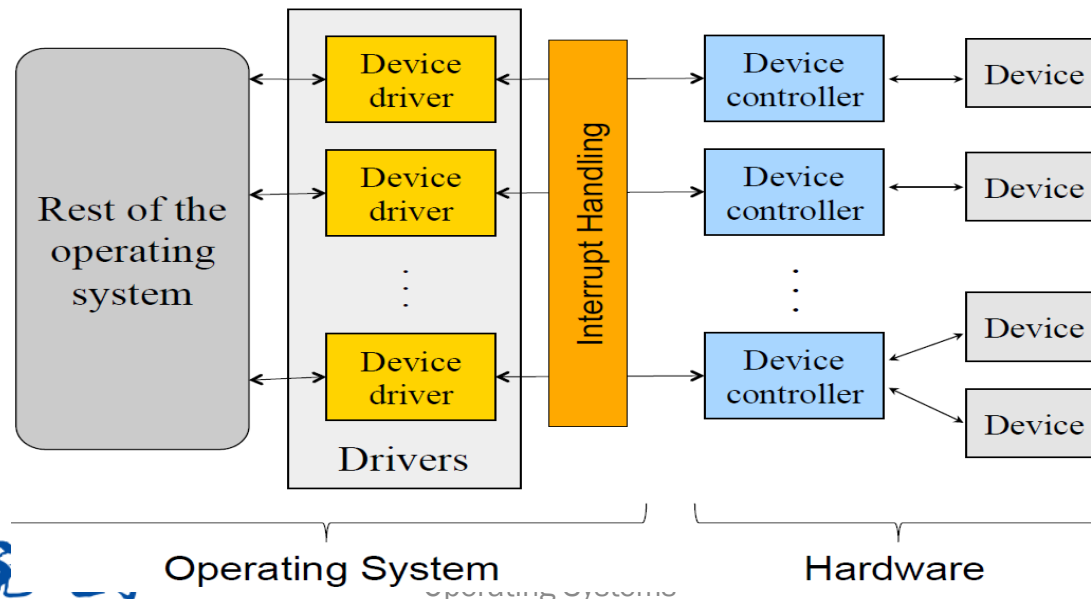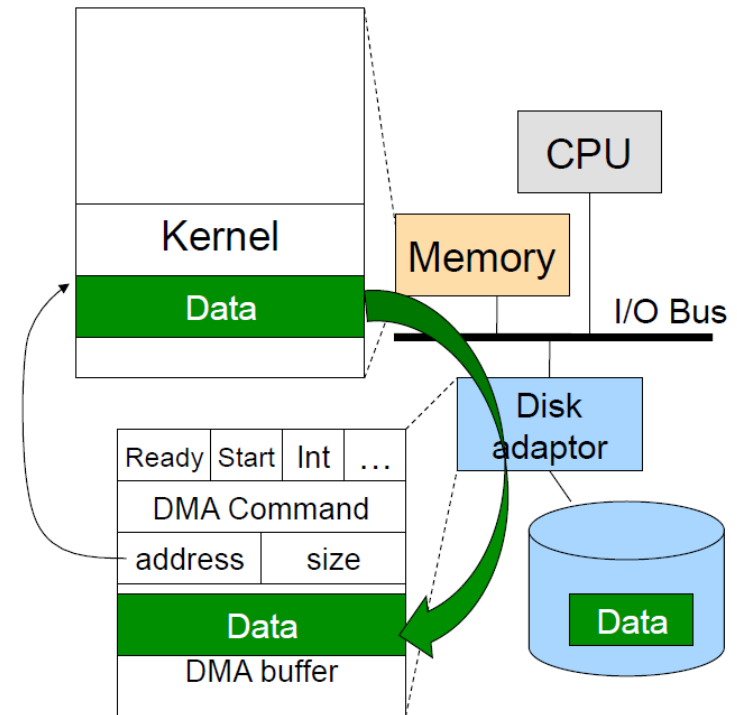
# Project 5 Device Driver

- MAC controller
  - Include two parts of registers
    - MAC registers
      - Base address 0xbfe1 0000
      - Totally 462 registers
    - DMA registers
      - Base address 0xbfe1 1000
      - Totally 22 registers
    - Accessing a register
      - Base address + $i$ * 4
      - $i$ indicates i[th] register ($i$ starts from 0)

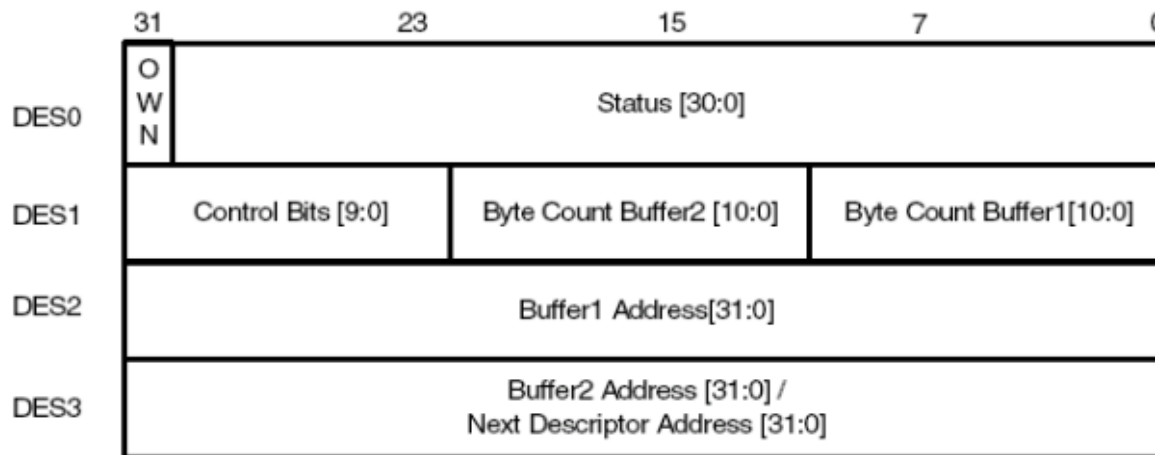| | 0xFFFF FFFF |
| Mapped | |
| | 0xC000 0000 |
| Unmapped Uncachable | |
| | 0xA000 0000 |
| Unmapped Cachable | |
| | 0x8000 0000 |
| Mapped | |
| | 0x0000 0000 |

University of Chinese Academy of Sciences

# Project 5 Device Driver

- Direct Memory Access
  - Exchange data between host and device bypassing CPU
  - DMA transmit descriptor
  - DMA receive descriptor

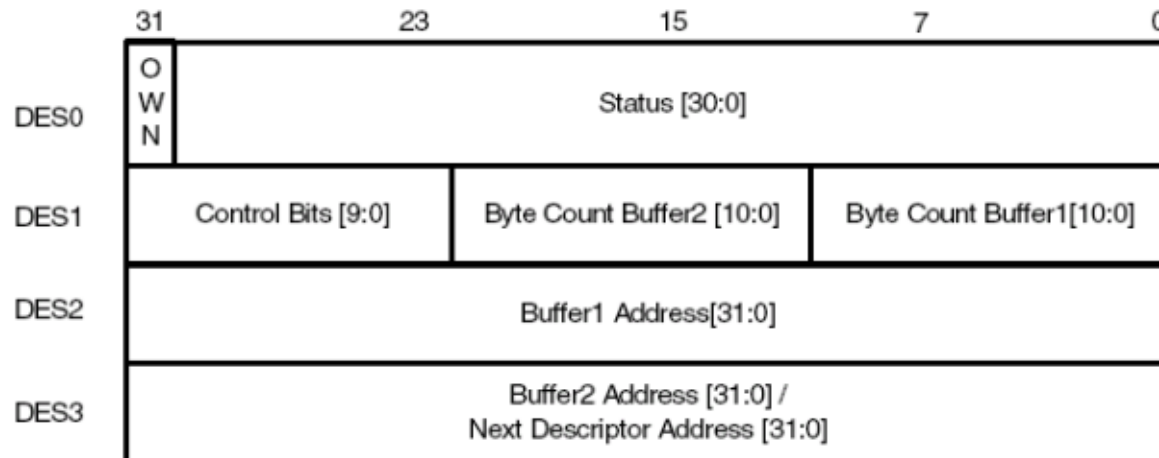University of Chinese Academy of Sciences

# Project 5 Device Driver

- DMA descriptor
  - 16 bytes
  - Transmit descriptor for transmitting data
  - Receive descriptor for receiving data
  - One descriptor for one network packet
    - You are required to handle 64 packets
    - A circle linked list is expected to used for multiple packets

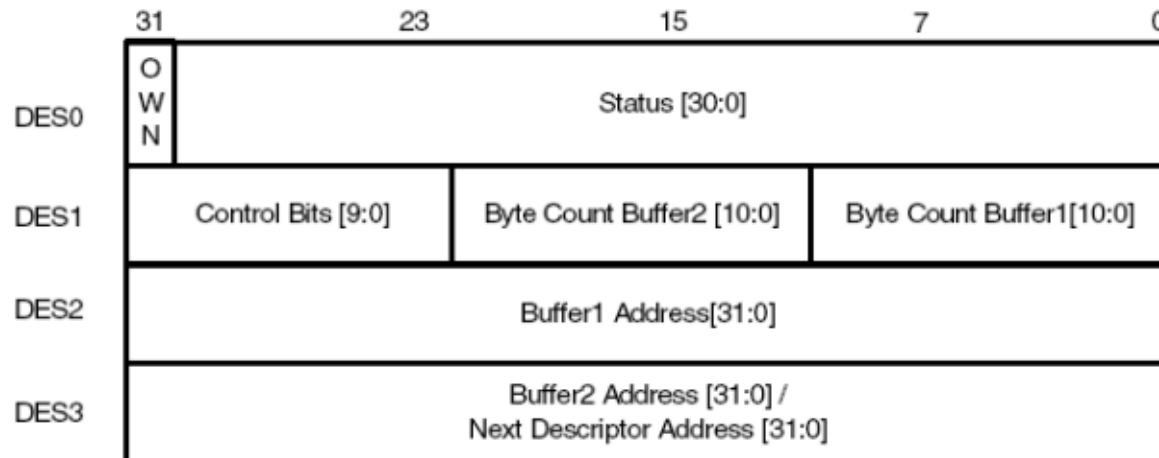| | 31 | | 23 | | 15 | | 7 | | 0 |
|---|---|---|---|---|---|---|---|---|---|
| DES0 | O W N | | | | Status [30:0] | | | | |
| DES1 | | Control Bits [9:0] | | Byte Count Buffer2 [10:0] | | | Byte Count Buffer1[10:0] | | |
| DES2 | | | | Buffer1 Address[31:0] | | | | | |
| DES3 | | | | Buffer2 Address [31:0] / Next Descriptor Address [31:0] | | | | | |

# Project 5 Device Driver

- DMA descriptor
  - OWN flag
    - 0 indicates host has the rights to access DMA buffer
    - 1 indicates DMA controller has the rights to access DMA buffer
    - Remember to set this bit to 1 to allow DMA controller to Tx/Rx data

| | 31 | 23 | 15 | 7 | 0 |
|---|---|---|---|---|---|
| DES0 | O W N | Status [30:0] | | | |
| DES1 | Control Bits [9:0] | | Byte Count Buffer2 [10:0] | Byte Count Buffer1[10:0] | |
| DES2 | Buffer1 Address[31:0] | | | | |
| DES3 | Buffer2 Address [31:0] / Next Descriptor Address [31:0] | | | | |

University of Chinese Academy of Sciences

# Project 5 Device Driver

- DMA descriptor
  - Status
    - Indicates different errors when transmitting or receiving data
    - Please refer to Fig. 4 and 7 for the meanings in the guiding book when you suffer from handling errors

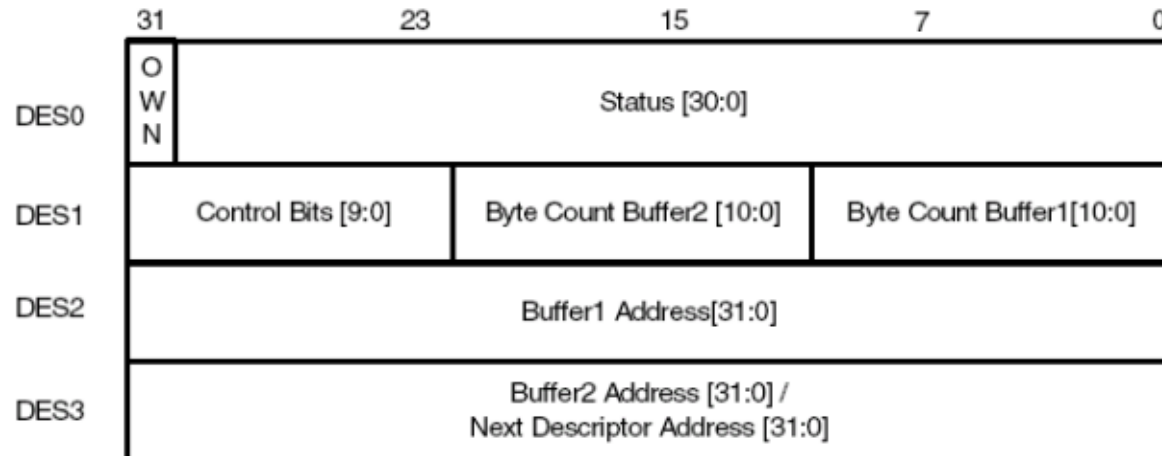| | 31 | 23 | 15 | 7 | 0 |
|---|---|---|---|---|---|
| DES0 | OWN | | Status [30:0] | | |
| DES1 | Control Bits [9:0] | | Byte Count Buffer2 [10:0] | Byte Count Buffer1[10:0] | |
| DES2 | Buffer1 Address[31:0] | | | | |
| DES3 | Buffer2 Address [31:0] / Next Descriptor Address [31:0] | | | | |

# Project 5 Device Driver

- DMA descriptor
  - Control Bits
    - TER/RER（25bit）: Transmit/Receive End of Ring
    - 1 indicates this descriptor is the end of a descriptor ring
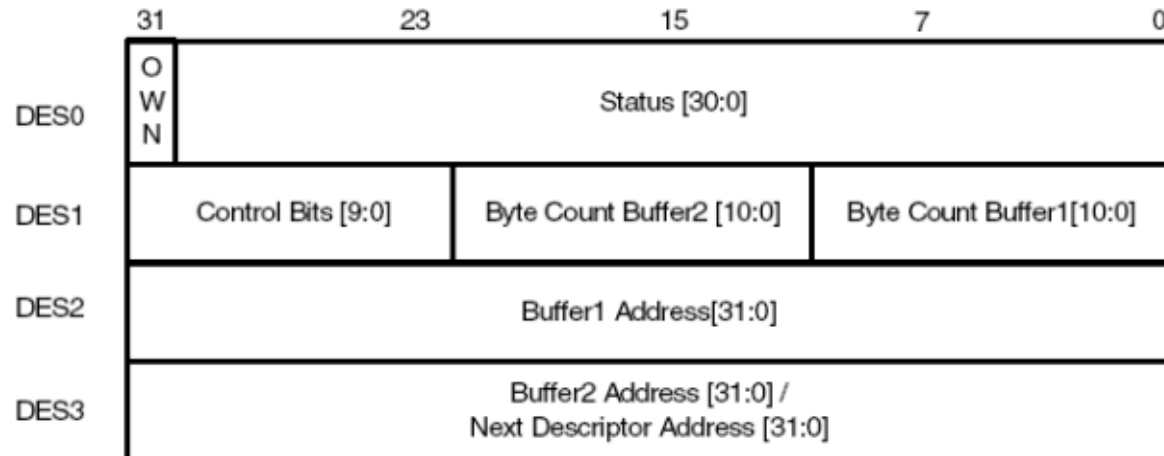    - Please set this bit when initializing the descriptor ring
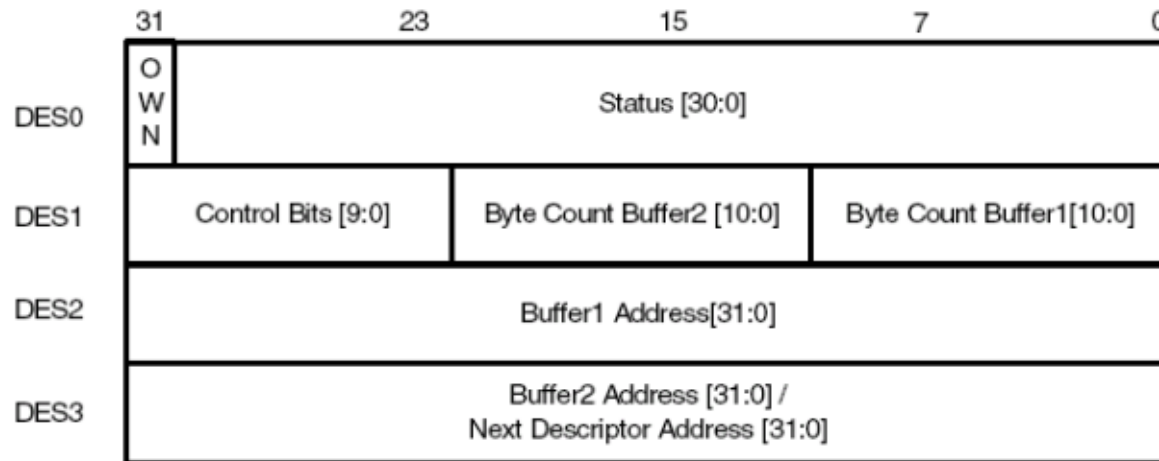
# Project 5 Device Driver

- ## DMA descriptor
  - ### Control Bits
    - TCH/RCH ( 24bit ) : Second Address Chained
    - 1 indicates the second buffer in this descriptor is linking the next descriptor
    - Please set this bit when initializing the descriptor ring

| | 31 | 23 | 15 | 7 | 0 |
|---|---|---|---|---|---|
| DES0 | O W N | | Status [30:0] | | |
| DES1 | Control Bits [9:0] | | Byte Count Buffer2 [10:0] | | Byte Count Buffer1[10:0] |
| DES2 | Buffer1 Address[31:0] | | | | |
| DES3 | Buffer2 Address [31:0] / Next Descriptor Address [31:0] | | | | |

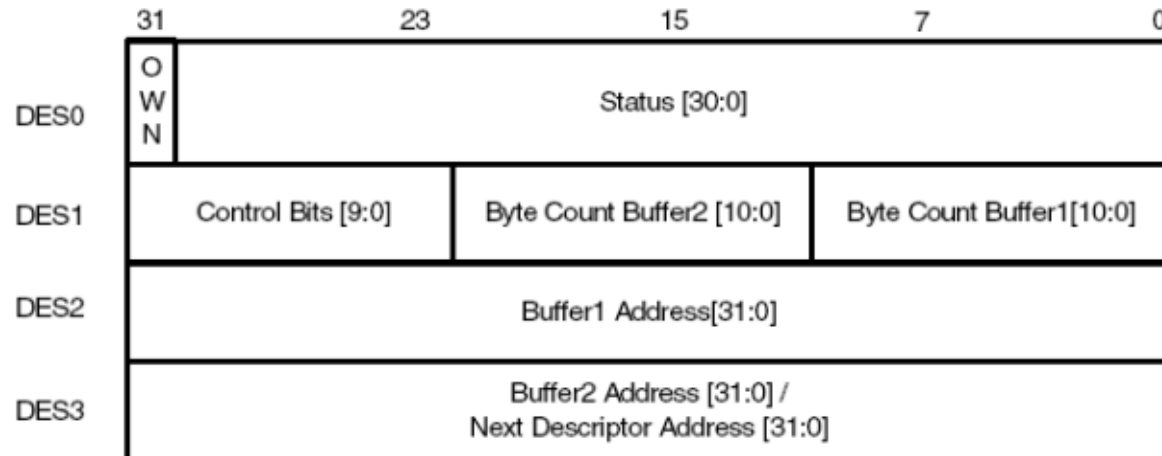University of Chinese Academy of Sciences

# Project 5 Device Driver

- DMA descriptor
  - Control Bits
    - Byte Count Buffer1: the size of buffer1 in bytes, note that we have 1KB data to transmit/receive when filling buffer1
    - Byte Count Buffer2: set to 0 in this project

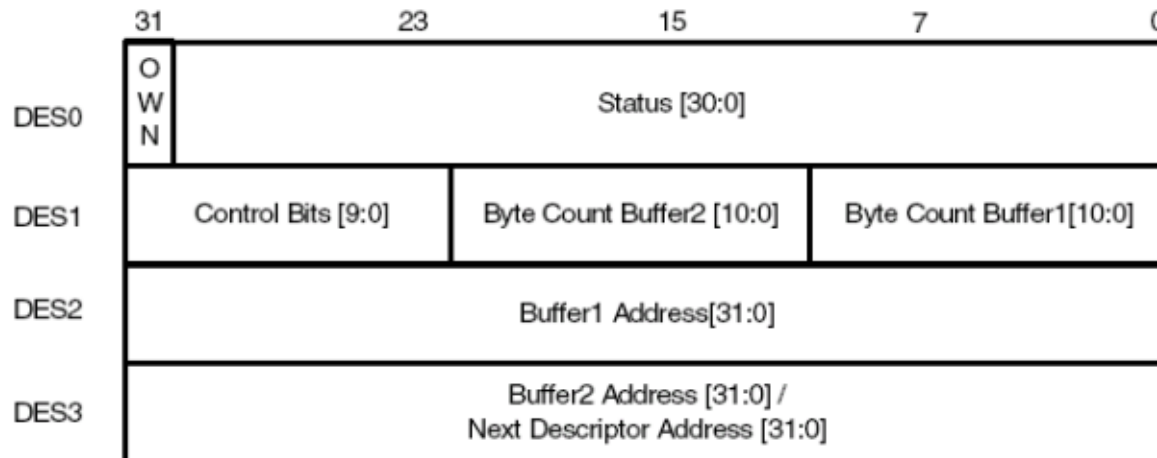| | 31 | 23 | 15 | 7 | 0 |
|---|---|---|---|---|---|
| DES0 | OWN | Status [30:0] | | | |
| DES1 | Control Bits [9:0] | | Byte Count Buffer2 [10:0] | | Byte Count Buffer1[10:0] |
| DES2 | Buffer1 Address[31:0] | | | | |
| DES3 | Buffer2 Address [31:0] / Next Descriptor Address [31:0] | | | | |

# Project 5 Device Driver

- ## DMA descriptor
  - ### Buffer1 Address
    - The memory address of the starting address of data buffer
    - Note that, please set the memory address in the unmapped area

| | 31 | | 23 | | 15 | | 7 | | 0 |
|---|---|---|---|---|---|---|---|---|---|
| DES0 | O W N | | | Status [30:0] | | | | | |
| DES1 | Control Bits [9:0] | | | Byte Count Buffer2 [10:0] | | | Byte Count Buffer1[10:0] | | |
| DES2 | Buffer1 Address[31:0] | | | | | | | | |
| DES3 | Buffer2 Address [31:0] / Next Descriptor Address [31:0] | | | | | | | | |

University of Chinese Academy of Sciences

# Project 5 Device Driver

- ## DMA descriptor
  - ### Buffer2 Address
    - The memory address of next descriptor
    - Note that, we have 64 packets to transmit/receive respectively, and each packet is 1KB which needs a descriptor

| | 31 | 23 | 15 | 7 | 0 |
|---|---|---|---|---|---|
| DES0 | OWN | Status [30:0] | | | |
| DES1 | Control Bits [9:0] | | Byte Count Buffer2 [10:0] | Byte Count Buffer1[10:0] | |
| DES2 | Buffer1 Address[31:0] | | | | |
| DES3 | Buffer2 Address [31:0] / Next Descriptor Address [31:0] | | | | |

University of Chinese Academy of Sciences

# Project 5 Device Driver

- Manipulating DMA descriptor
  - After filling DMA descriptor
    - You need to set the DMA register 4 and DMA register 3 to be the starting addresses of TX descriptor and RX descriptor, respectively
  - Before sending a packet
    - You need to write DMA register 1 with any value to trigger transmission
  - Before receiving a packet
    - You need to write DMA register 2 with any value to trigger receiving

University of Chinese Academy of Sciences
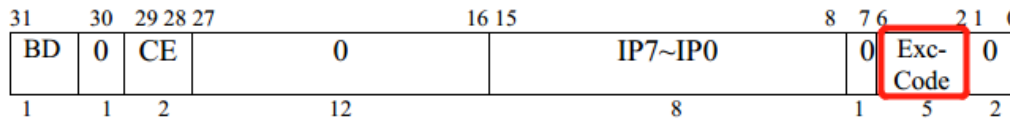
# Project 5 Device Driver

- Network packet
  - We already set the contents of each packet
    - The first 12 bytes refer to Ethernet frame head
    - The following 20 bytes refer to IP head
    - Then the following 8 bytes refer to UDP head
    - The rest refer to data payload
    - Please do not change the packet contents UNLESS you are sure what you do

```
uint32_t buffer[PSIZE] = {0xffffffff, 0x5500ffff, 0xf77db57b,
0x00450008, 0x0000d400, 0x11ff0040, 0xa8c073d8, 0x00e00101,
0xe914fb00, 0x0004e914, 0x0000, 0x005e0001, 0x2300fb00,
0x84b7f28b, 0x00450008, 0x0000d400, 0x11ff0040, 0xa8c073d8,
0x00e00101, 0xe914fb00, 0x0801e914, 0x0000};
```

# Project 5 Device Driver

- MAC interrupt handler
  - Cause register
    - Exc-code: 0 refers to interrupt
    - IP3: 1 indicates device interrupt

| 31 | 30 | 29 28 27 | 16 15 | 8 7 6 | 2 1 0 |
|----|----|----------|-------|-------|-------|
| BD | 0 | CE | 0 | IP7~IP0 | 0 | Exc-Code | 0 |
| 1 | 1 | 2 | 12 | 8 | 1 | 5 | 2 |

# Project 5 Device Driver

- MAC interrupt handler
  - 5 groups of registers correspond to device interrupts
  - Each group has 6 registers
    - INT$X$_SR
    - INT$X$_EN
    - INT$X$_SET
    - INT$X$_CLR
    - INT$X$_POL
    - INT$X$_EDGE

# Project 5 Device Driver

- MAC interrupt handler
  - INT$X$_SR indicates the interrupt type (X is from 0 to 4)
    - e.g. INT0_SR refers to the first 32 types of interrupt
  - No. of MAC interrupt is 35
    - MAC interrupt is indicated by bit 3 in INT1_SR
    - MAC interrupt is enabled by bit 3 in INT1_EN
    - Please refer guiding book to understand how to set these bits using the register group

# Project 5 Device Driver

- MAC interrupt handler
  - INT1_SR register
    - Address: 0xbfd01058
    - bit 3:  1 indicates MAC interrupt
  - INT1_EN register
    - Address: 0xbfd0105c
    - bit 3: 1 enables MAC interrupt

University of  Chinese Academy of Sciences

# Project 5 Device Driver

- MAC interrupt handler
  - Check DMA RX descriptors to receive incoming data
  - Add this interrupt handler into *interrupt_helper*
  - <span style="color:red">Note that, you need to set corresponding IM bits when initializing interrupt</span>

University of Chinese Academy of Sciences

# Project 5 Device Driver

- Step by step
  - Task 1: Setup TX/RX DMA descriptors
    - Initialize DMA descriptors for transmitting and receiving data
    - Allocating data buffer for transmission and receiving, and fill given packets into transmit buffer
    - Set corresponding DMA registers to allow transmitting and receiving data
    - Implement syscalls for above functions
    - <span style="color:red">Note that, in this task, the receive thread continuously checks whether there is incoming data</span>
    - <span style="color:red">We do not have hard requirement for the upper bound of the number of packets in this task, but at least you need to receive 10 packets and send 64 packets</span>

University of Chinese Academy of Sciences

# Project 5 Device Driver

- Step by step
  - Task 2: implement MAC driver with blocking mode
    - Base on task1, but the receive thread blocks itself when there is no incoming data. The receive thread is unblocked until it receives 64 packets
    - The receive thread checks the incoming data in timer interrupt
    - The transmit thread does not change

University of Chinese Academy of Sciences
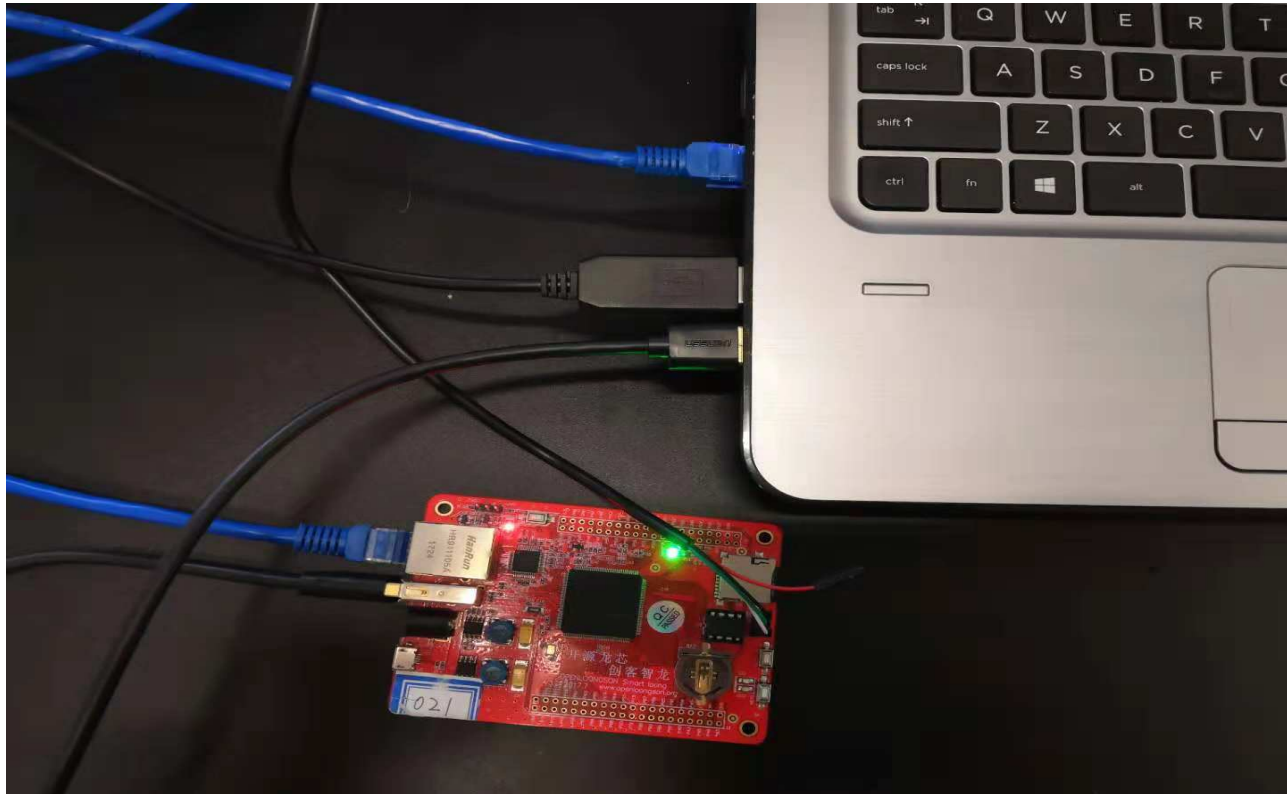
# Project 5 Device Driver

- Step by step
  - Task 3: handling MAC interrupt
    - Initialize and enable MAC interrupt
    - Implement MAC interrupt handler for receiving data
    - The receive thread checks the number of packets during the MAC interrupt handler until it receives 64 packets.
    - The transmit thread does not change

University of Chinese Academy of Sciences

# Project 5 Device Driver

- Task testing

# Project 5 Device Driver

- Task testing
  - For transmitting data
    - Use wireshark (Windows) or tcpdump (Linux) to examine if the transmit thread successfully transmit data
    - Please read our reference documents
  - For receiving data
    - Use pktRxTx to transmit data from host to your development card to test receiving data

University of Chinese Academy of Sciences

# Project 5 Device Driver

- Requirements for design review (40 points)
  - What are the memory addresses of TX/RX descriptors? Please give an example about the contents of filled TX/RX descriptors? Which registers do you use to handle DMA process?
  - How many packets can you receive in task 1?
  - What is the procedure for receiving data in task 2?
  - What is the procedure of your MAC interrupt handler?
  - What is the procedure for transmitting data?

# Project 5 Device Driver

- Requirements of developing (60 points)
  - Setup TX/RX descriptor and allow transmitting and receiving data (25)
  - Implement receiving data with blocking mode (15)
  - Implement MAC interrupt handler (20)

# Project 5 Device Driver

- Bonus: receive large amount of data (2 points)
  - Use your MAC interrupt handler to receive large amount of data
  - pktTxRx can send data at 1MB/s ~ 2MB/s, let us send data for e.g. 10 seconds
  - Test the bandwidth of your receive thread, the expected bandwidth should be at least 1Mbit/s
  - Note that
    - your MAC interrupt handler should be robust enough to handle large amount of packets in a short time
    - You may need to handle packet loss

# Project 5 Device Driver

- P5 schedule
  - P5 design review: 9$^{th}$ Dec.
  - Scheduled P5 due: 16$^{th}$ Dec.

University of Chinese Academy of Sciences