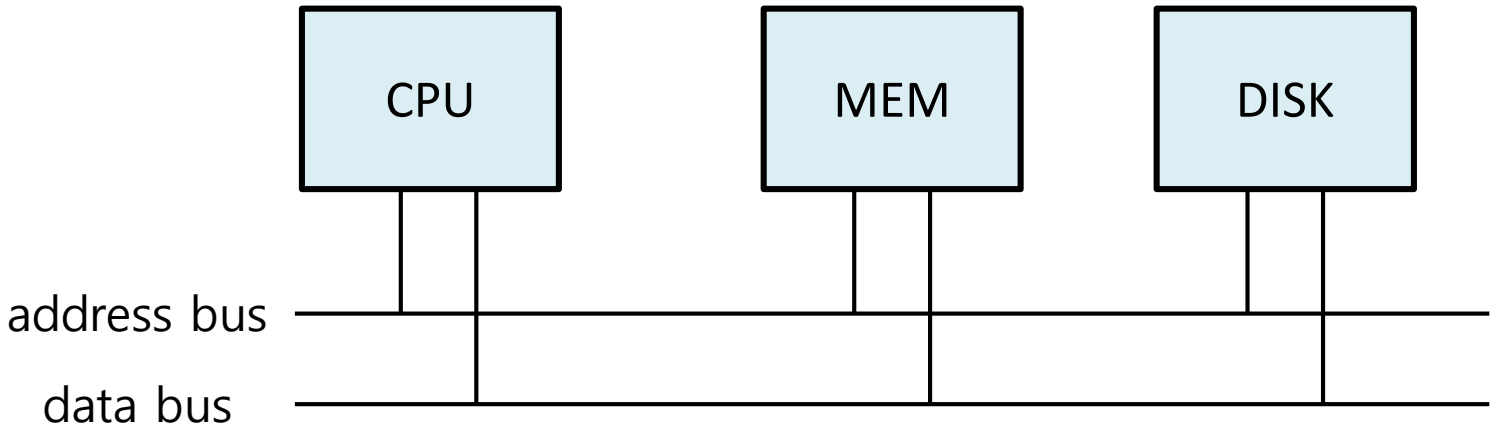
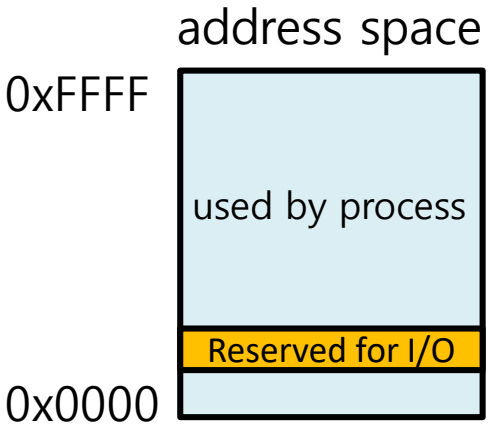
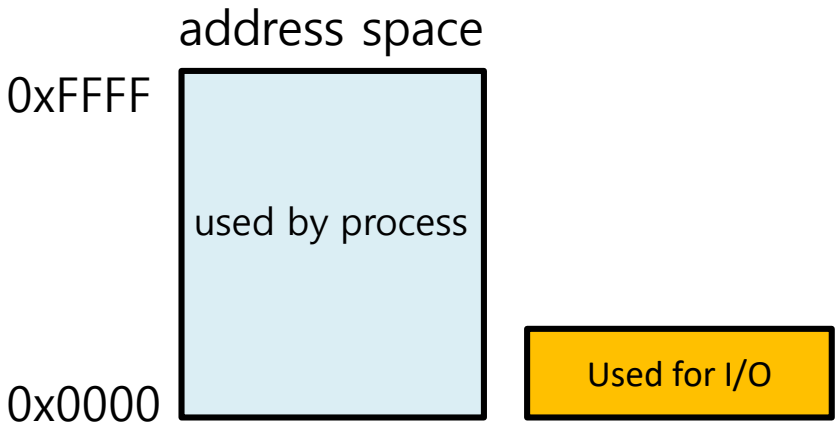
Lab: memory-mapped I/O



- CPU communicates with other devices including memory, peripherals through bus.
  - Address bus for selecting device
  - Data bus for transferring data

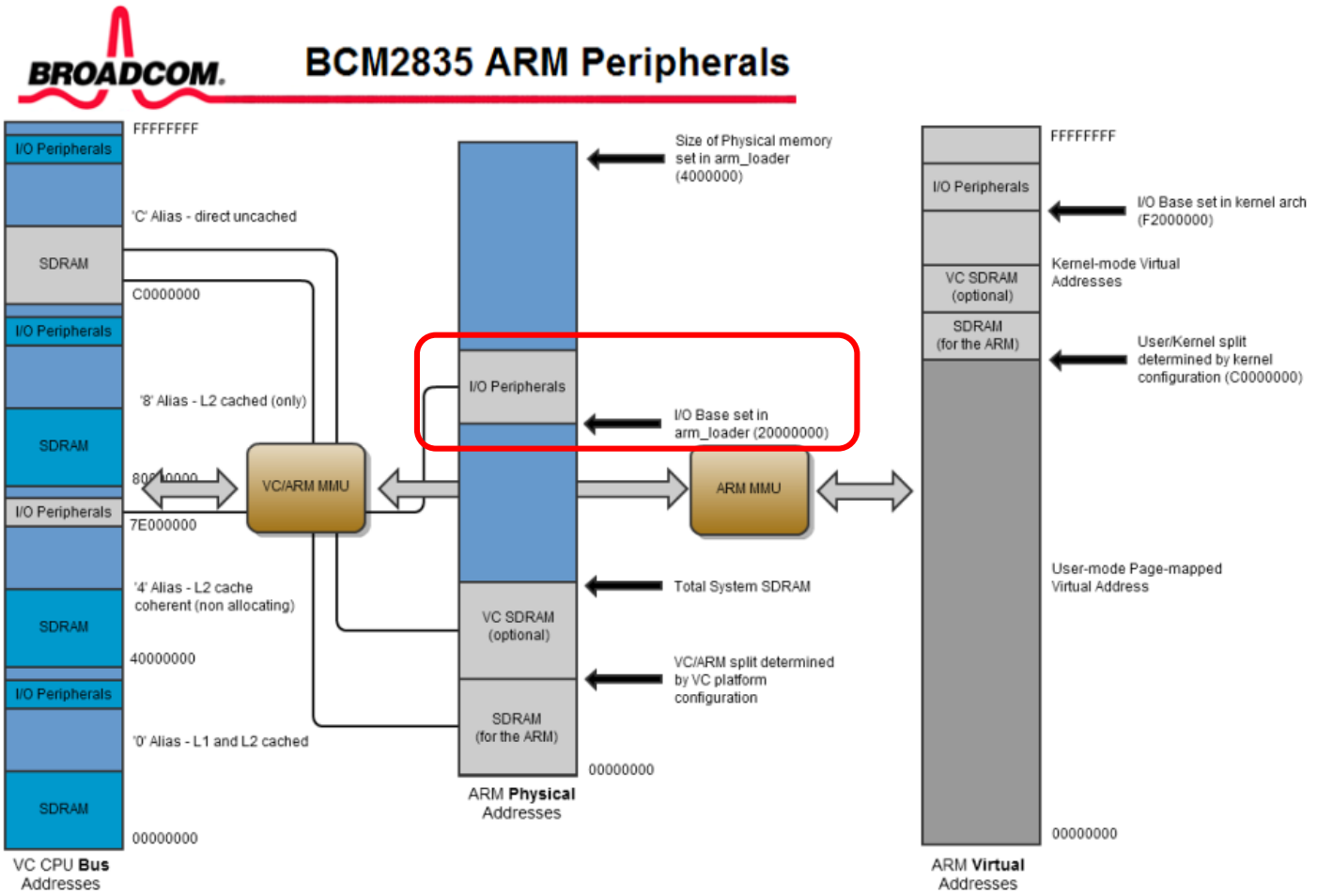
Lab: memory-mapped I/O





- Two complementary method of performing I/O between the CPU and devices.
  - Memory-mapped I/O (MMIO)
    - uses the same address space to address both main memory and I/O devices. The memory and registers of the I/O devices are mapped to (associated with) address values, so a memory address may refer to either a portion of physical RAM or to memory and registers of the I/O device.
  - Port-mapped I/O(PMIO), (referred as Isolated I/O)
    - uses a special class of CPU instructions designed specifically for performing I/O, such as the *in* and *out* instructions found on microprocessors based on the x86 architecture. I/O devices have a separate address space from general memory.

Lab: memory-mapped I/O



Bus address **Physical address** **Virtual address**

Reference: BCM2711 ARM Peripherals, Broadcom Corporation
BCM2835 ARM Peripherals, Broadcom Corporation

Lab: memory-mapped I/O

- RaspberryPI uses MMIO
  - I/O peripherals are reserved to use 0x20000000 of the physical address.
    - NOTE: the reserved address is determined as **0xFE000000 in raspberry PI 4** (BCM2711)
  - And, this address will be mapped to the 0x7E000000 of the bus address.

- Bus address
  - A special address space used by the BCM series processor to access the devices connected to the bus, including memory and I/O peripherals.
  - The ARM physical memory address will be translated to the bus address by the MMU.

Lab: controlling GPIO

The GPIO has the following registers. All accesses are assumed to be 32-bit. The GPIO register base address is `0x7e200000`.

Offset	Name	Description
0x00	GPFSEL0	GPIO Function Select 0
0x04	GPFSEL1	GPIO Function Select 1
0x08	GPFSEL2	GPIO Function Select 2
0x0c	GPFSEL3	GPIO Function Select 3
0x10	GPFSEL4	GPIO Function Select 4
0x14	GPFSEL5	GPIO Function Select 5
0x1c	GPSET0	GPIO Pin Output Set 0
0x20	GPSET1	GPIO Pin Output Set 1
0x28	GPCLR0	GPIO Pin Output Clear 0
0x2c	GPCLR1	GPIO Pin Output Clear 1
0x34	GPLEV0	GPIO Pin Level 0
0x38	GPLEV1	GPIO Pin Level 1
0x40	GPEDS0	GPIO Pin Event Detect Status 0
0x44	GPEDS1	GPIO Pin Event Detect Status 1

GPFSEL0 Register

Description

The function select registers are used to define the operation of the general-purpose I/O pins. Each of the 58 GPIO pins has at least two alternative functions as defined in Section 5.3. The FSEL*n* field determines the functionality of the *n*th GPIO pin. All unused alternative function lines are tied to ground and will output a "0" if selected. All pins reset to normal GPIO input operation.

Bits	Name	Description	Type	Reset
31:30	Reserved.	-	-	-
29:27	FSEL9	FSEL9 - Function Select 9 000 = GPIO Pin 9 is an input 001 = GPIO Pin 9 is an output 100 = GPIO Pin 9 takes alternate function 0 101 = GPIO Pin 9 takes alternate function 1 110 = GPIO Pin 9 takes alternate function 2 111 = GPIO Pin 9 takes alternate function 3 011 = GPIO Pin 9 takes alternate function 4 010 = GPIO Pin 9 takes alternate function 5	RW	0x0
26:24	FSEL8	FSEL8 - Function Select 8	RW	0x0
23:21	FSEL7	FSEL7 - Function Select 7	RW	0x0
20:18	FSEL6	FSEL6 - Function Select 6	RW	0x0
17:15	FSEL5	FSEL5 - Function Select 5	RW	0x0
14:12	FSEL4	FSEL4 - Function Select 4	RW	0x0
11:9	FSEL3	FSEL3 - Function Select 3	RW	0x0
8:6	FSEL2	FSEL2 - Function Select 2	RW	0x0
5:3	FSEL1	FSEL1 - Function Select 1	RW	0x0
2:0	FSEL0	FSEL0 - Function Select 0	RW	0x0

Lab: controlling GPIO

GPSET0 Register

Description

The output set registers are used to set a GPIO pin. The SET*n* field defines the respective GPIO pin to set, writing a "0" to the field has no effect. If the GPIO pin is being used as an input (by default) then the value in the SET*n* field is ignored. However, if the pin is subsequently defined as an output then the bit will be set according to the last set/clear operation. Separating the set and clear functions removes the need for read-modify-write operations

Bits	Name	Description	Type	Reset
31:0	SET*n* (*n*=0..31)	0 = No effect 1 = Set GPIO pin *n*	WO	0x00000000

GPSET1 Register

Bits	Name	Description	Type	Reset
31:26	Reserved.	-	-	-
25:0	SET*n* (*n*=32..57)	0 = No effect 1 = Set GPIO pin *n*.	WO	0x0000000

Lab: controlling GPIO

GPCLR0 Register

Description

The output clear registers are used to clear a GPIO pin. The CLR*n* field defines the respective GPIO pin to clear, writing a "0" to the field has no effect. If the GPIO pin is being used as an input (by default) then the value in the CLR*n* field is ignored. However, if the pin is subsequently defined as an output then the bit will be set according to the last set/clear operation. Separating the set and clear functions removes the need for read-modify-write operations.

Bits	Name	Description	Type	Reset
31:0	CLR*n* (*n*=0..31)	0 = No effect 1 = Clear GPIO pin *n*	WO	0x00000000

GPCLR1 Register

Bits	Name	Description	Type	Reset
31:26	Reserved.	-	-	-
25:0	CLR*n* (*n*=32..57)	0 = No effect 1 = Clear GPIO pin *n*	WO	0x0000000

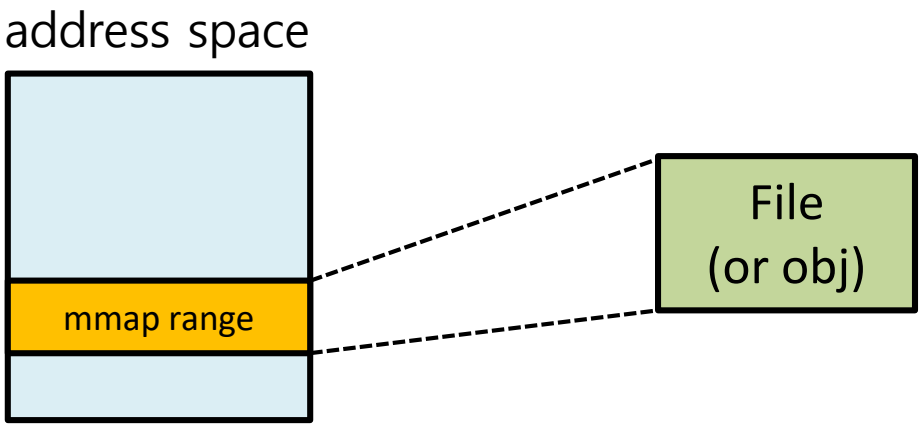
Lab: controlling GPIO

- GPFSEL0: select GPIO pin function
- GPSET: set GPIO value as 1
- GPCLR: clear GPIO value as 0

Lab: controlling GPIO

- Mmap
  - mmap() creates a new mapping in the virtual address space of the calling process.
  - prototype

```
#include <sys/mman.h>

void *mmap(void addr[.length], size_t length, int prot, int flags,
                    int fd, off_t offset);
int munmap(void addr[.length], size_t length);
```



Lab: controlling GPIO

```c
#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/mman.h>

#define BCM_IO_BASE 0xFE000000 /* Only for RPI4 */

#define GPIO_BASE (BCM_IO_BASE + 0x200000)
#define GPIO_SIZE (256) /* 0x7E2000B0 - 0x7E2000000 + 4 = 176 + 4 = 180 */

volatile unsigned *gpio; /* For MMIO */

int main(int argc, char **argv)
{
    int gno, i, mem_fd;
    void *gpio_map;

    if(argc < 2) {
        printf("Usage : %s GPIO_NO\n", argv[0]);
        return -1;
    }
```

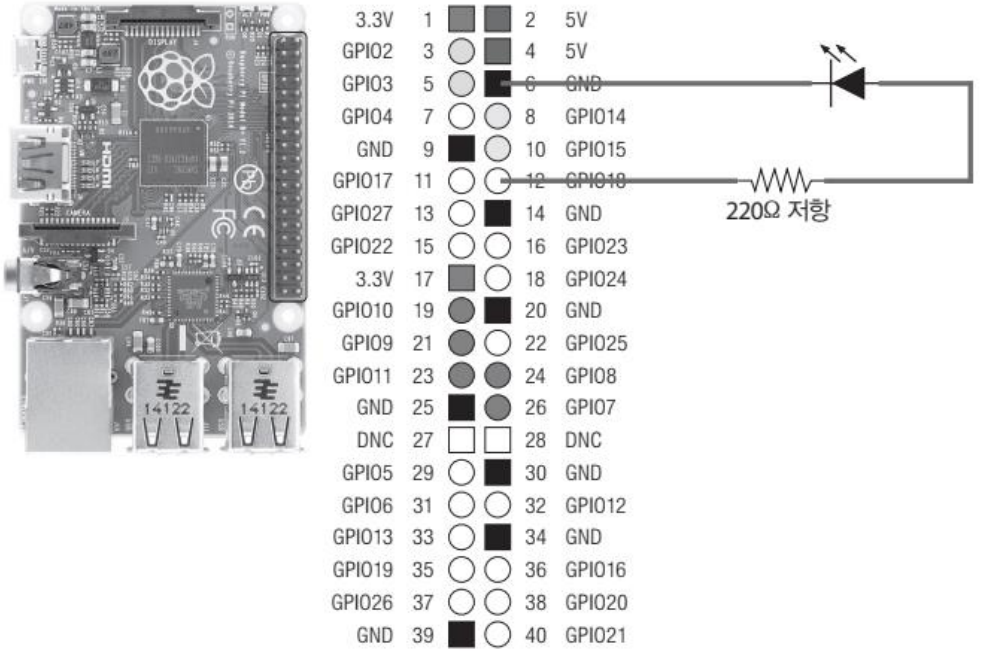
Lab: controlling GPIO

Contd.

```
    gno = atoi(argv[1]);

    /* /dev/mem provides a physical memory layout */
    if ((mem_fd = open("/dev/mem", O_RDWR | O_SYNC) ) < 0) {
        perror("open() /dev/mem\n");
        return -1;
    }

    /* GPIO and mmap */
    gpio_map = mmap(NULL, GPIO_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, mem_fd, GPIO_BASE);
    if (gpio_map == MAP_FAILED) {
        printf("[Error] mmap() : %d\n", (int)gpio_map);
        perror -1;
    }

    gpio = (volatile unsigned *)gpio_map;

    /* Put Your I/O code here (Blink the LED 5 times)
       Hint: 1. Set the pin mode as output
             2. Set the pin value as 1 for a sec
             3. Clear the pin value as 0 for a sec
     */

    munmap(gpio_map, GPIO_SIZE);
    close(mem_fd);

return 0;
}
```

Lab: controlling GPIO

- Set the circuit
  - Use the GPIO18 pin.



- Build & run

```
pi@raspberrypi:~ $ gcc -o mm mm_gpio.c
pi@raspberrypi:~ $ sudo ./mm 18
```