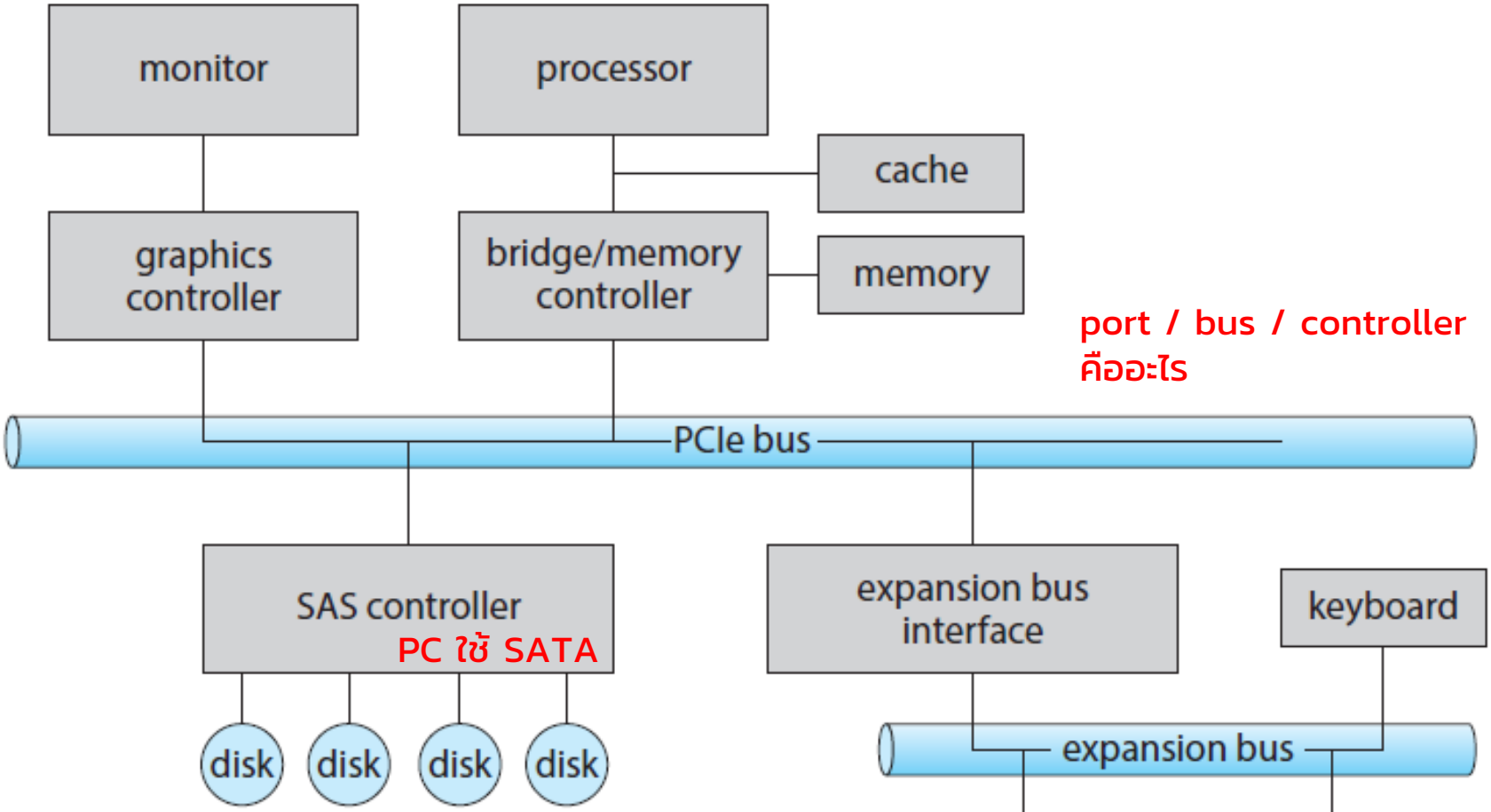
แก้ไขครั้งล่าสุดเมื่อวันที่ 30 ตุลาคม 2567

I/O Systems



port / bus / controller คืออะไร

SAS = Serial Attached SCSI

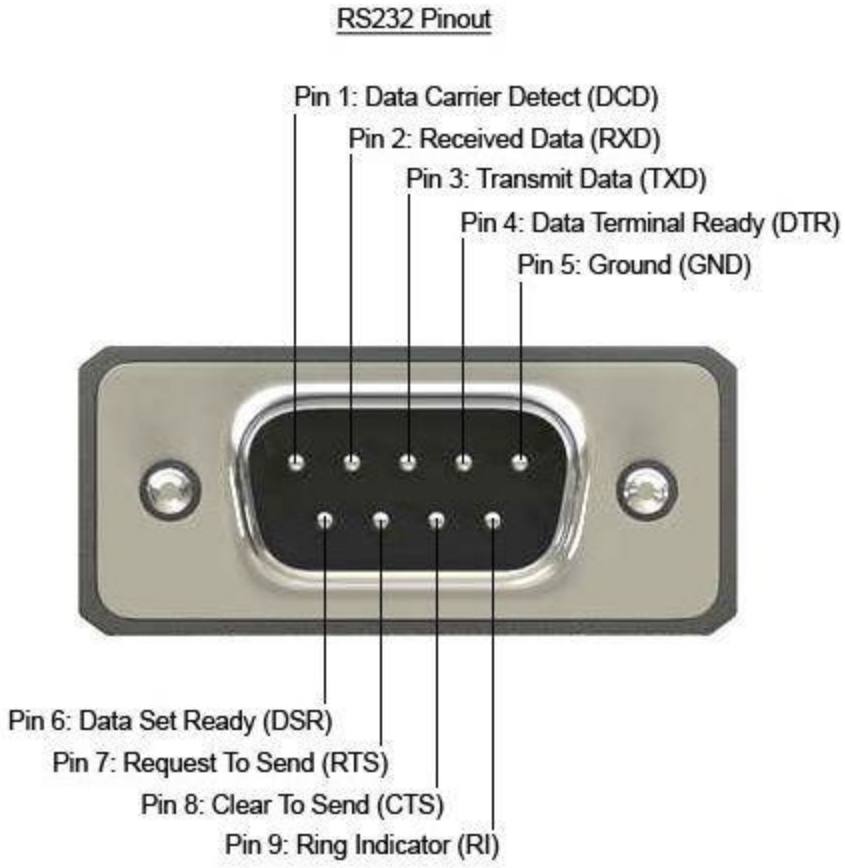
SATA = Serial ATA

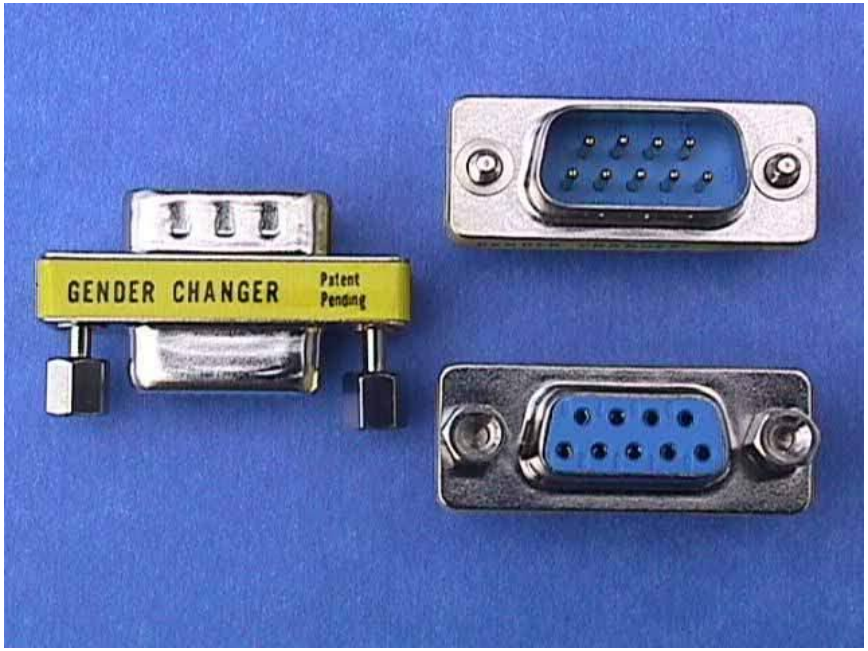
ATA = Advanced Technology Attachment

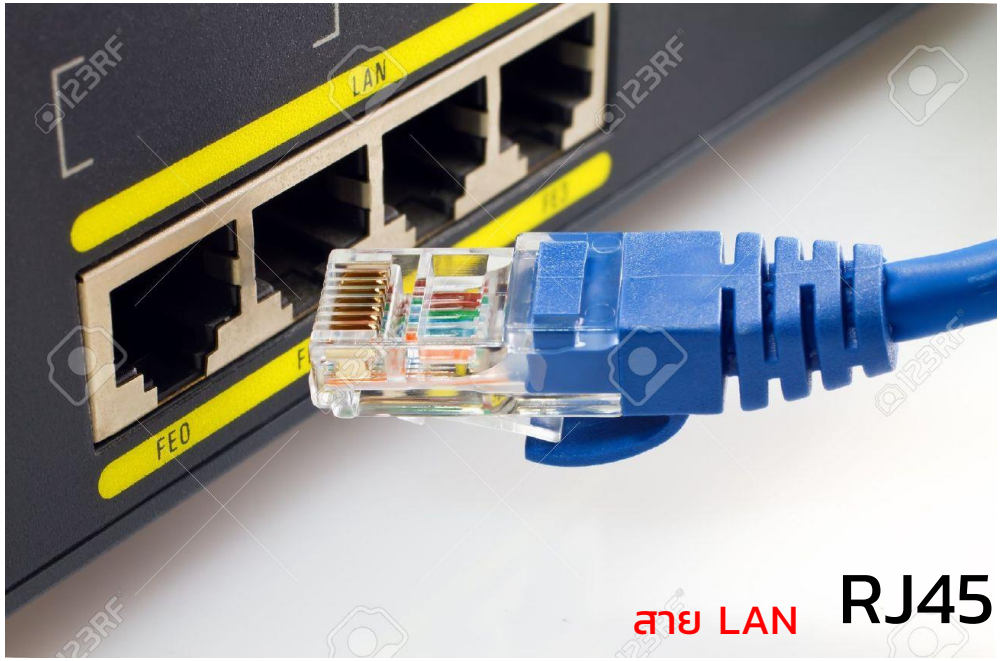
Figure 12.1 A typical PC bus structure.

Port

The device communicates with the machine via a connection point, or port – for example, a serial port.



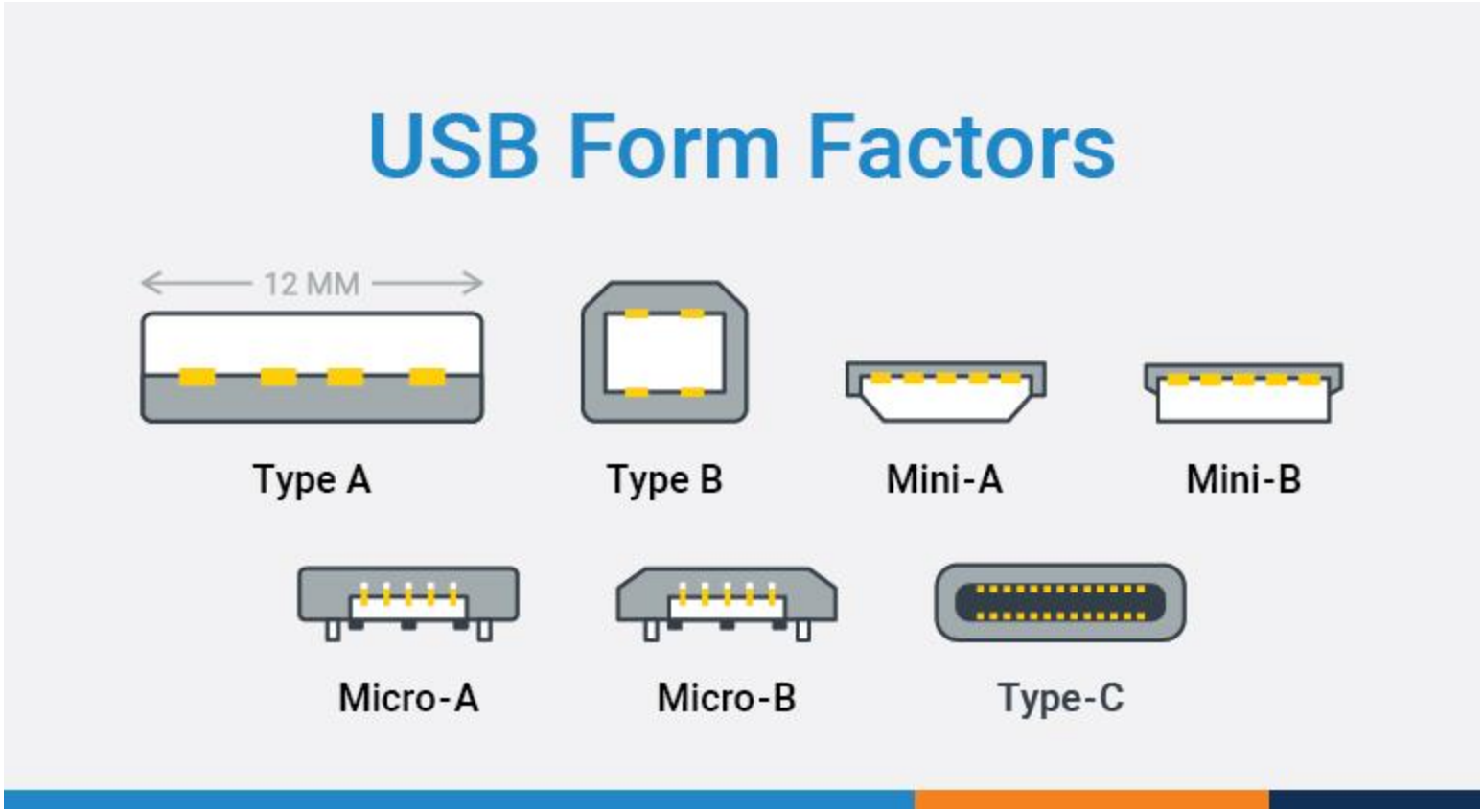




สาย LAN **RJ45**

RJ11

สายโทรศัพท์



Type-A

Type-B

Mini-B

Micro-B

Type-C

Thunderbolt 3 (40 Gbps/s)

Thunderbolt 4 (40 Gbps/s)

port เหมือน USB-C ทำงานร่วมกับ USB-C device ได้
แต่เอา Thunderbolt device ไปต่อกับ USB-C port ไม่ได้

พอร์ต Thunderbolt เดิมทีเป็นพอร์ตความเร็วสูงที่เกิดขึ้นในปี ค.ศ. 2011 (พ.ศ. 2554) จากความร่วมมือในการพัฒนาระหว่าง บริษัท Apple และ Intel ทำให้ในช่วงแรกของการพัฒนา พอร์ต Thunderbolt จะมีให้ใช้แค่ในอุปกรณ์ของ Apple เท่านั้น

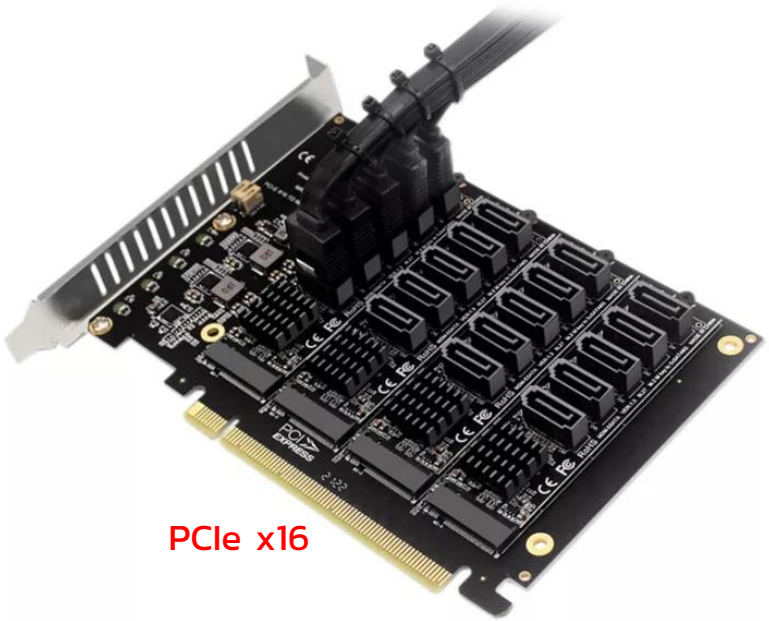
Bus wires + protocol

A bus is a set of wires and a rigidly defined protocol that specifies a set of messages that can be sent on the wires.

PCI = Peripheral Component Interconnect
PCIe = PCI express

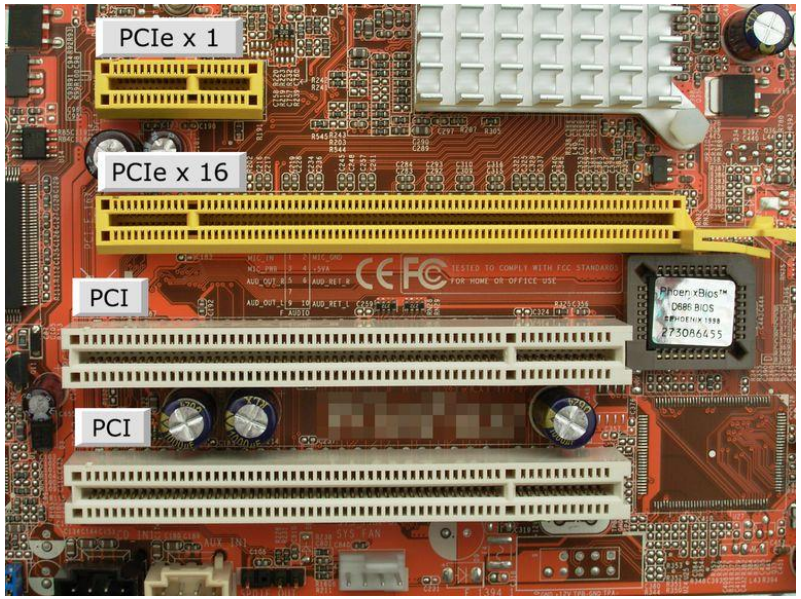
Expansion bus สำหรับอุปกรณ์ที่ทำงานไม่เร็วมาก เช่น คีย์บอร์ด เมาส์ USB devices

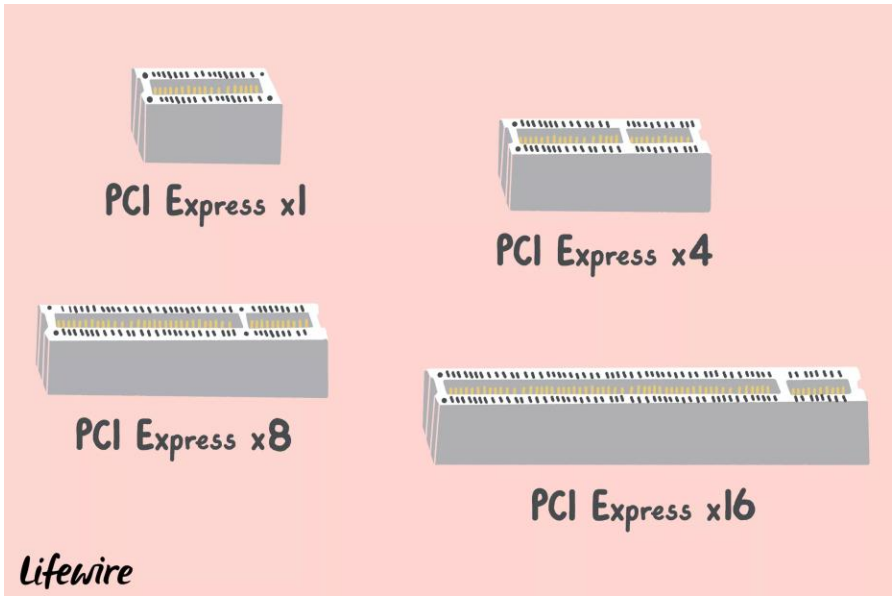
Graphic card



SATA card

PCI (Peripheral Component Interconnect) bus

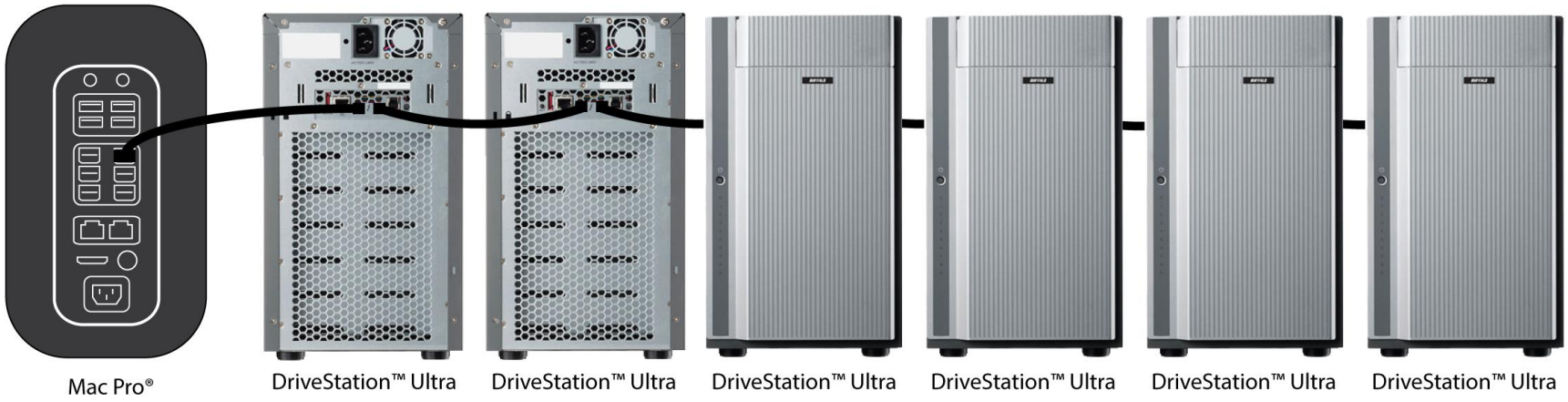


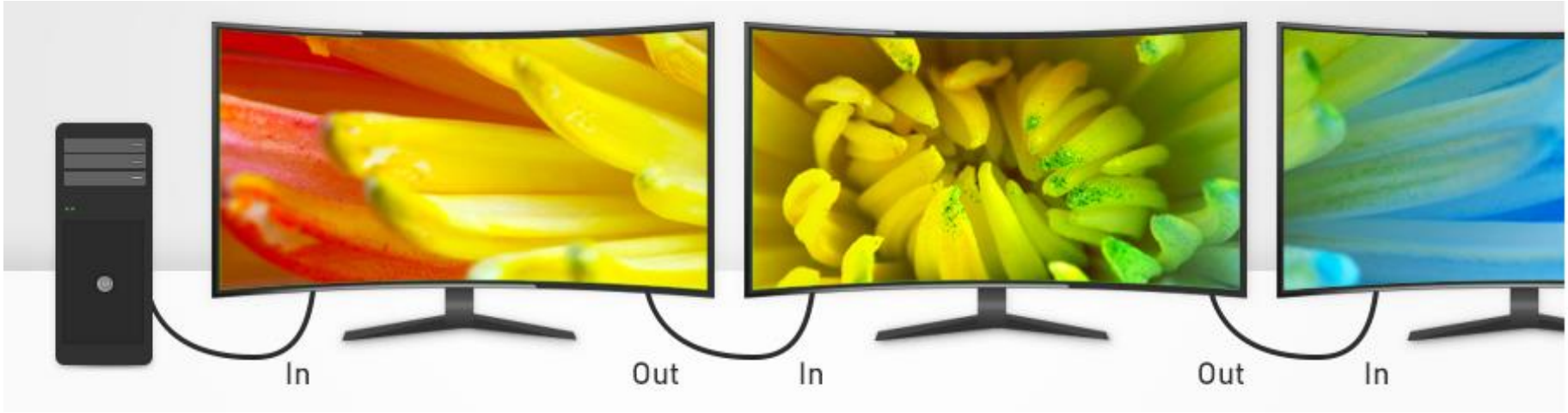


ปัจจุบันน่าจะเหลือแค่แบบ Express

PCIe x 1

Daisy-Chaining



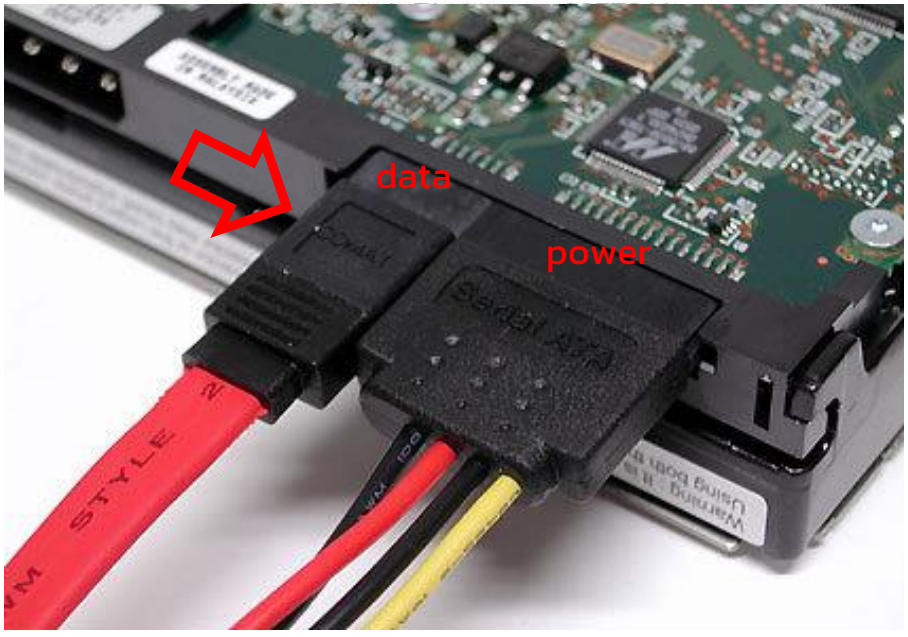


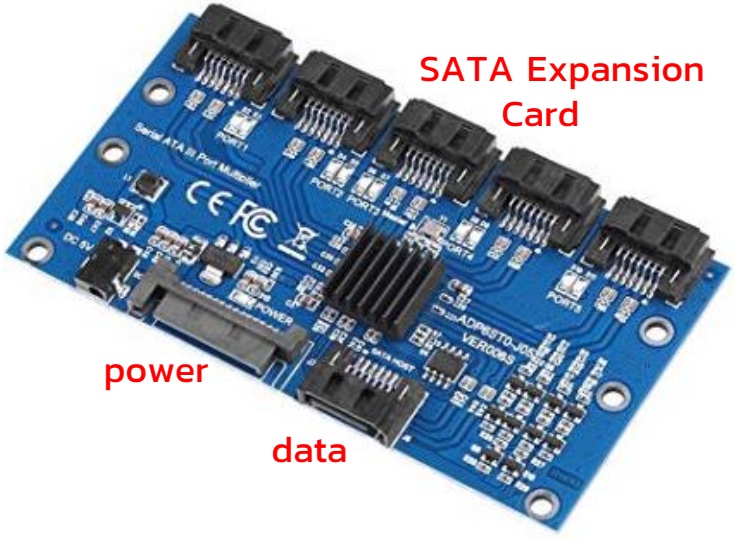
Controller

A controller is a collection of electronics that can operate a port, a bus, or a device.

SATA

SATA Hub

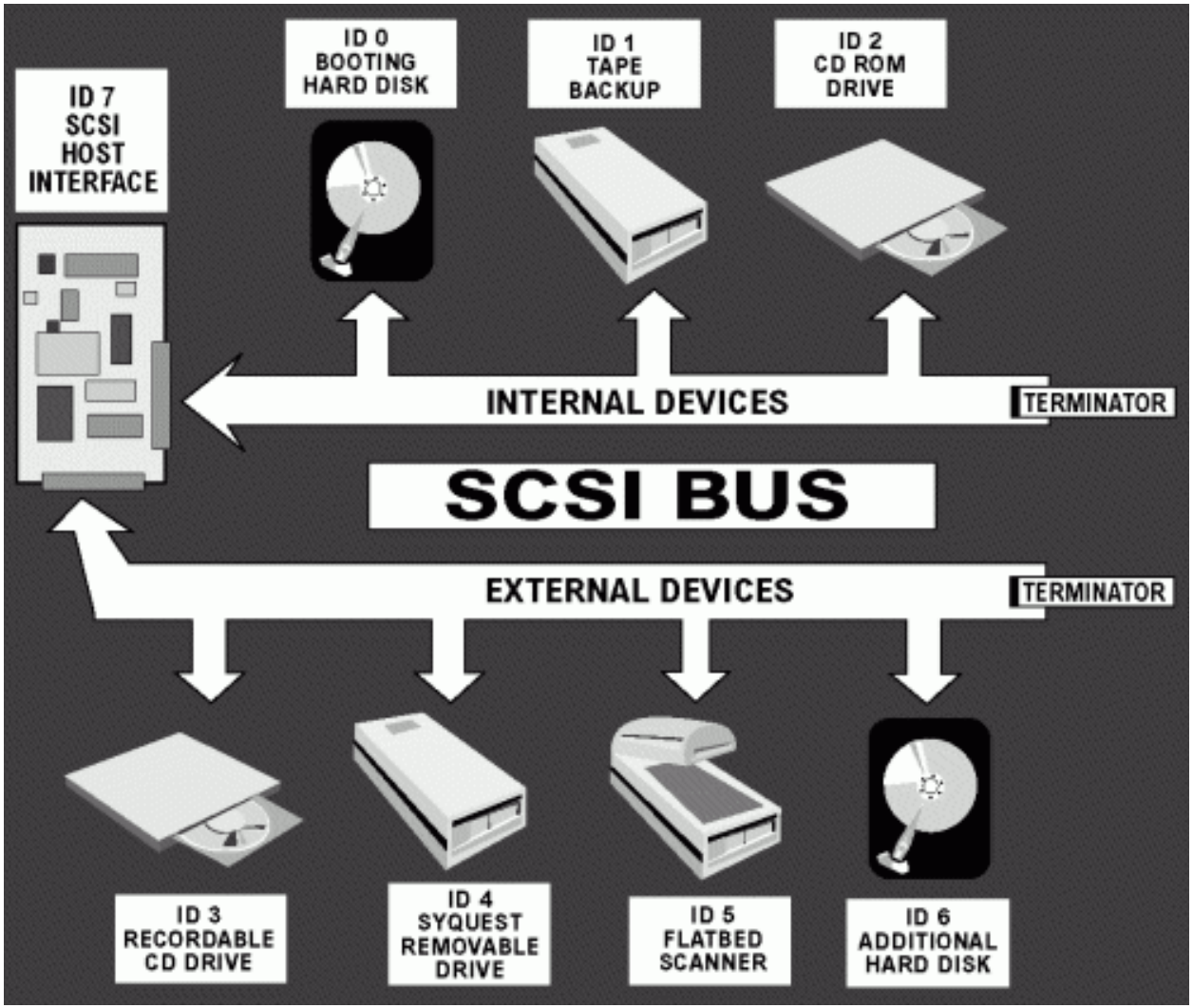




SATA Expansion Card

SCSI

มีหลาย device บน bus ได้ ไม่ใช่แบบ point-to-point, เป็นแบบ parallel ไม่ใช่ serial



ไม่ต้องใช้ hub หรือ expansion card

https://www.pctechguide.com/interfaces/scsi-explained-with-pictures

SAS = Serial attached SCSI

a point-to-point serial protocol that moves data to and from computer-storage devices such as hard disk drives and tape drives. SAS replaces the older Parallel SCSI (Parallel Small Computer System Interface.

Serial Attached SCSI

SAS connector

Width in bits	1
No. of devices	65,535 ใช้ expander
Speed	SAS-1: Full-duplex[1] 3 Gbit/s (2004) SAS-2: Full-duplex 6 Gbit/s (2009) SAS-3: Full-duplex 12 Gbit/s (2013) SAS-4: Full-duplex 22.5 Gbit/s (2017)[2]
Style	Serial
Hotplugging interface	Yes

History [edit]

- SAS-1: 3.0 Gbit/s, introduced in 2004[7]
- SAS-2: 6.0 Gbit/s, available since February 2009
- SAS-3: 12.0 Gbit/s, available since March 2013
- SAS-4: 22.5 Gbit/s called "24G",[8] standard completed in 2017[7][2]
- SAS-5: 45 Gbit/s is being developed[9]

Storage servers housing 24 SAS hard disk drives per server

https://en.wikipedia.org/wiki/Serial_Attached_SCSI

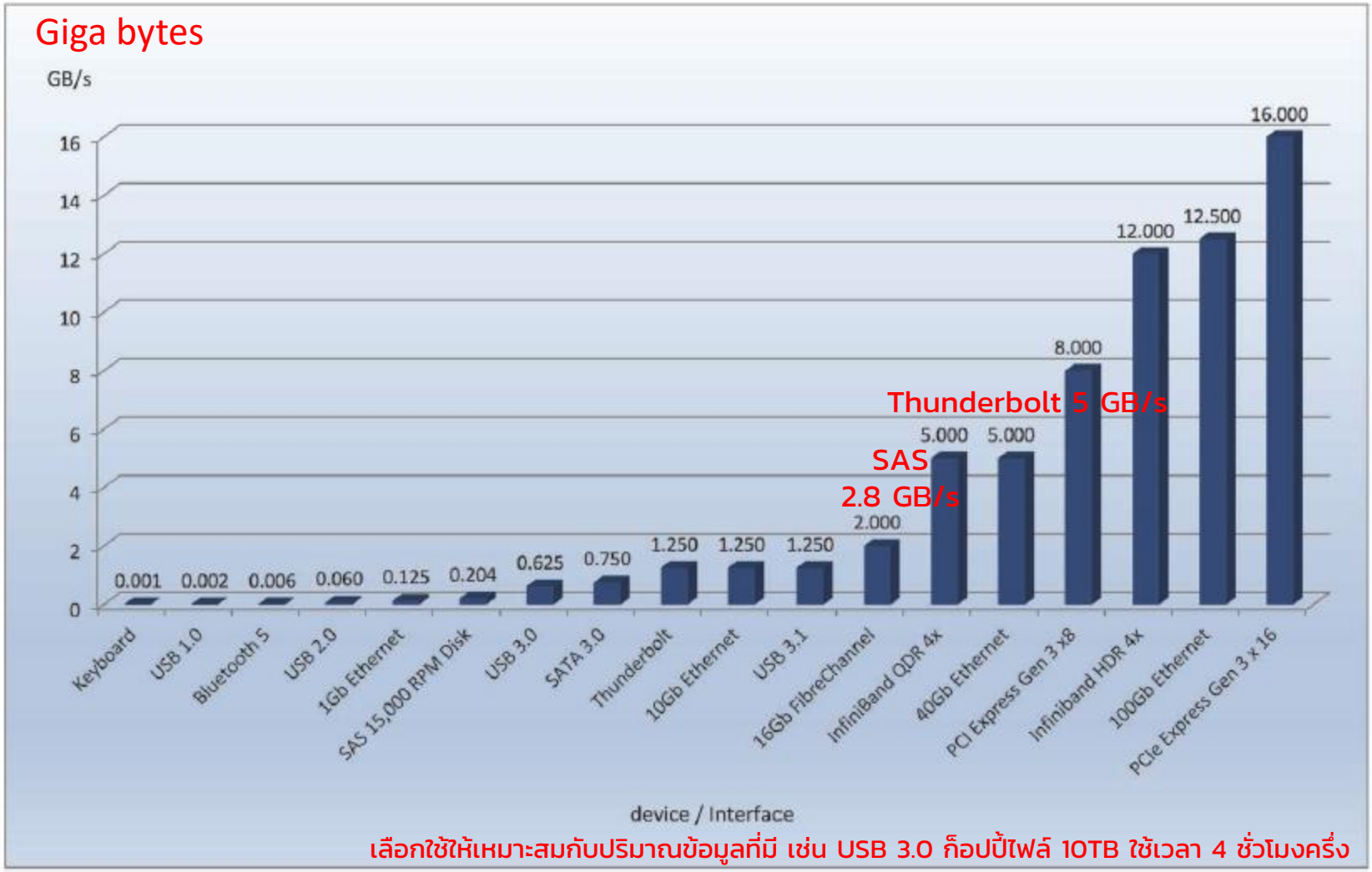


Figure 12.11 Common PC and data-center I/O device and interface speeds.

https://community.fs.com/article/10gbase-t-vs-sfp-which-one-is-the-best-option-for-10gbe-data-center-cabling.html

Synchronous / Asynchronous

Blocking

Non-blocking

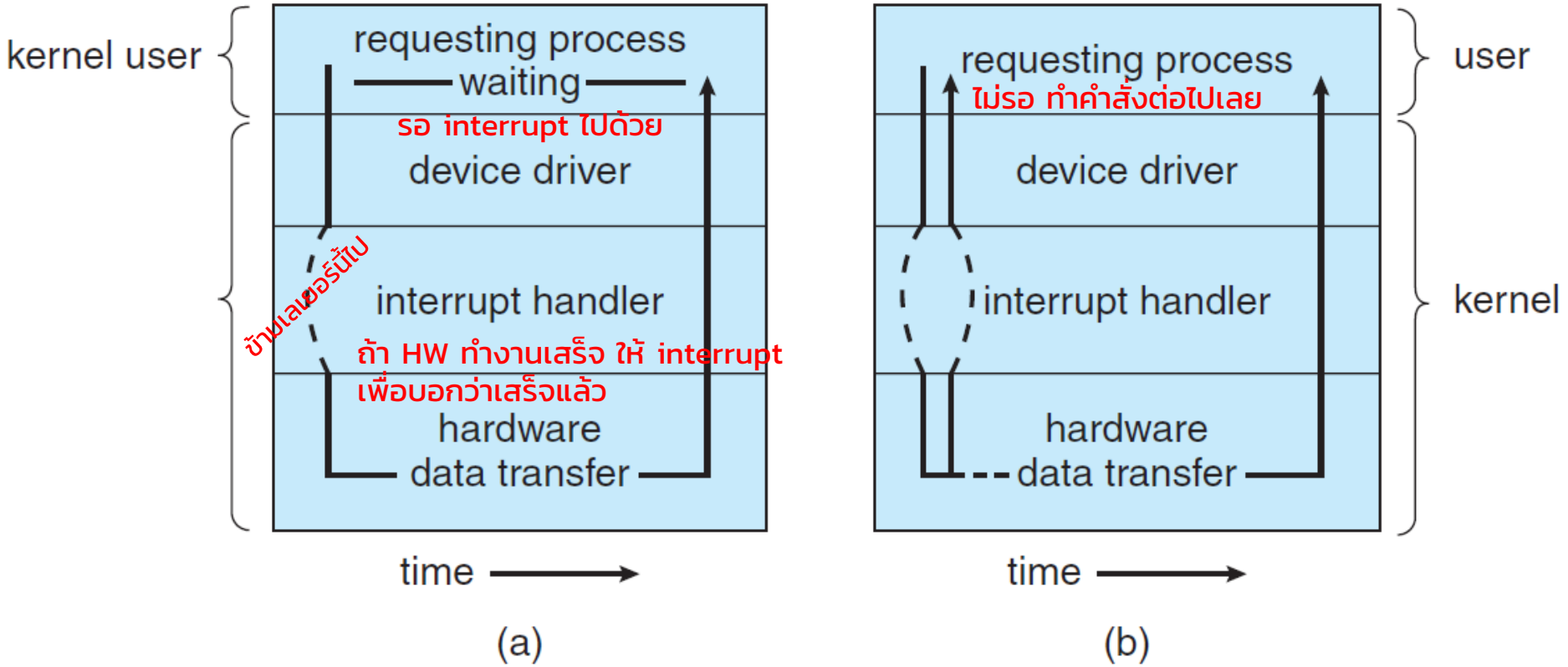
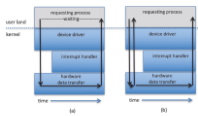


Figure 13.8 Two I/O methods: (a) synchronous and (b) asynchronous.



C# (IO-bound process, library มี asynchronous method ให้ใช้)

ไม่ต้องสร้าง thead

```
static void Main(string[] args) {

    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < 10000000; i++) sb.Append("1234567890\n");

    StreamWriter wt = new StreamWriter("data.txt");

    wt.WriteLineAsync(sb.ToString());
    // do other things, don't wait writing

    wt.Close();

    return;
}
```

เกิด error เนื่องจาก Close() ก่อนที่ WriteLineAsync() จะทำงานเสร็จ
ถ้าใส่ try ไว้ จะเห็นว่า Close() ไม่ได้ เพราะ write ยังไม่เสร็จ

```
Unhandled exception. System.InvalidOperationException: The stream is currently in use by a previous operation on the stream.
   at System.IO.StreamWriter.ThrowAsyncIOInProgress()
   at System.IO.StreamWriter.Dispose(Boolean disposing)
   at System.IO.StreamWriter.Close()
```

อ่านเพิ่มเติมเรื่อง promise, async/await
https://javascript.info/async-await

JavaScript (promise, then, catch)

```
1  const p = new Promise((resolve,reject) => {
2
3    /*
4     * do something
5     */
6
7    resolve('success')
8    //reject('error')
9  });
10
11 p.then((result) => {
12   console.log(result)
13 }).catch((error) => {
14   console.log(error);
15 });
```

Task (thread)

JavaScript เป็น single-threaded environment threads จะต้องต่อคิวกันเพื่อรอใช้ CPU ถ้า thread ข้างหน้า block เช่น ดาวน์โหลดไฟล์ใหญ่มาก thread ข้างหลังที่อัปเดต UI จะไม่ได้ใช้ CPU ทำให้ UI ค้าง

เช่น ดาวน์โหลด แบบ async

เมื่อดาวน์โหลดเสร็จแล้วจะ interrupt ให้ทำ then / catch

อัปเดต UI

อัปเดต UI

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise/then

C# (CPU-bound process, แบ่งงานเป็น task ย่อยหลาย ๆ task เอง)

บน multi-threaded environment
asynchronous tasks จะถูกแปลงเป็น threads ให้อัตโนมัติ ไม่จำเป็นต้อง 1 ต่อ 1
OS จะพิจารณาสร้าง threads จาก #cores และ workload ในขณะนั้น

Parameter (input)

```
private class Param {

    public int Start;
    public int End;

    public Param(int start, int end) {

        this.Start = start;
        this.End = end;
    }
}
```

Task (asynchronous)

```
static Sum DoSomethingAsync(Param p) {

    int value = 0;
    for (int i = p.Start; i <= p.End; i++) {

        value += i;
    }
    สมมติว่าใช้เวลา 1 วินาที
    Thread.Sleep(1 * 1000);

    return new Sum(value);
}
```

Result (output)

```
private class Sum {

    public int Value;

    public Sum(int value) {

        this.Value = value;
    }
}
```

```
static async Task Main(string[] args) { // if there is a result, return Task<...> instead of Task
    List<Task<Sum>> Tasks = new List<Task<Sum>>();

    for (int i = 11; i <= 20; i++) {

        Param p = new Param(1, i);

        // The variable p is needed (passing by reference)
        // DoSomethingAsync(new Param(1, i)) is a bug because
        // the value of i will be changed (i = 16) when running the task
        Task<Sum> t = Task.Run(() => DoSomethingAsync(p));
        Tasks.Add(t);
    }

    int total = 0;
    while (Tasks.Count > 0) {

        Task<Sum> task = (Task<Sum>)(await Task.WhenAny(Tasks));
        total += task.Result.Value;
        Tasks.Remove(task);
    }

    Console.WriteLine(total);

    return;
}
```

ถ้ามีการใช้ await ก็ต้องเติม async

สร้าง task มา 10 tasks แล้วไปทำงานอื่นต่อทันที (ถ้ามี) ไม่รอให้ tasks เสร็จ framework จะสร้าง thread ให้เอง

เก็บผลลัพธ์จากแต่ละ task มารวมกัน

ใช้เวลารัน 1 วินาทีเศษก็เสร็จ เพราะใช้ multi cores/threads

Chapter 12 I/O Systems