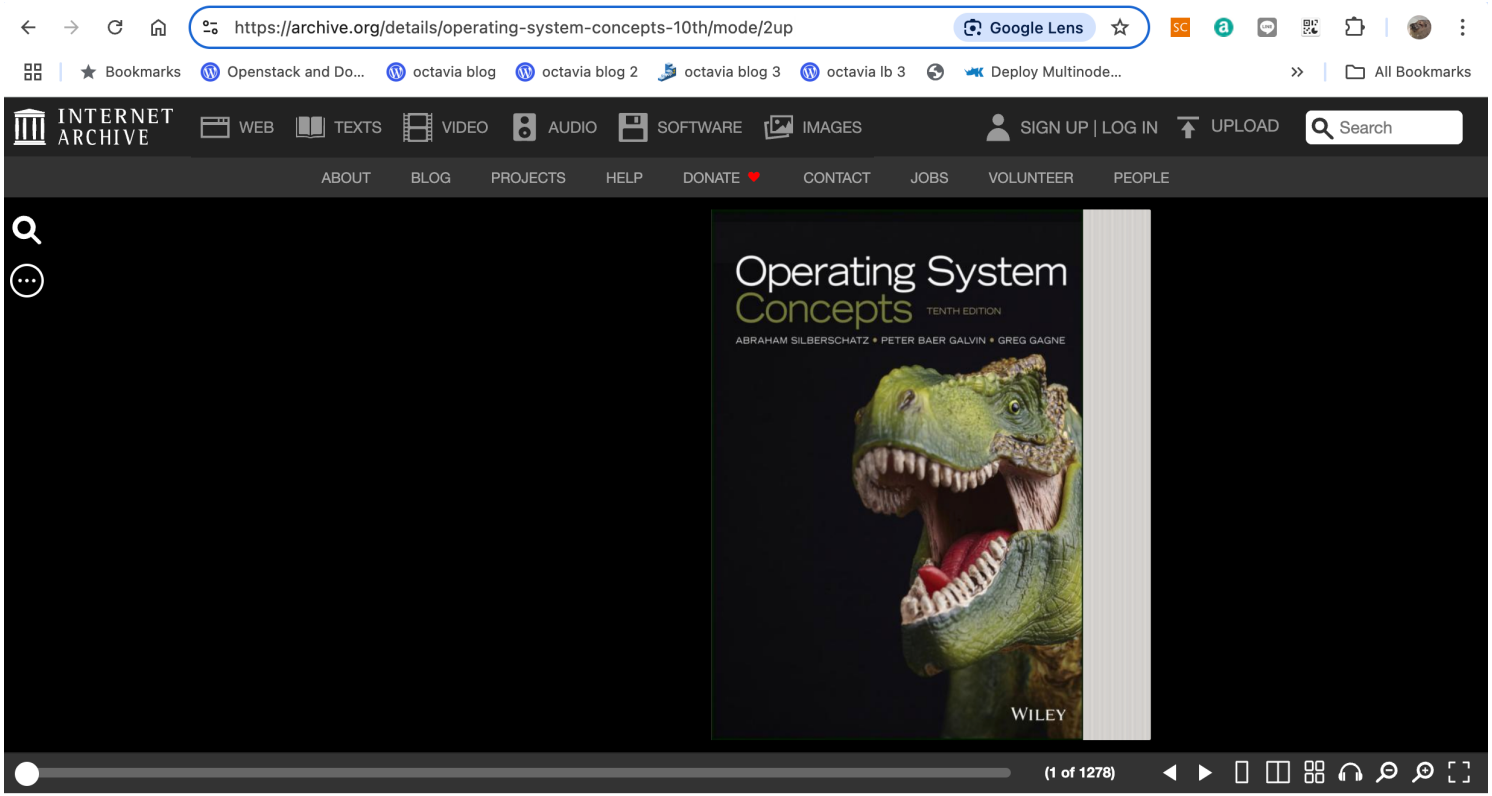
CS222
Operating Systems

Lecture 02

(Section 100001)
ผศ. ดร. กษิดิศ ชาญเชี่ยว
ckasidit@tu.ac.th

Textbook



- https://archive.org/details/operating-system-concepts-10th/mode/2up
- Original Slides
- https://www.os-book.com/OS10/slide-dir/index.html

Chapter 1: Introduction

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Operations
- Resource Management
- Security and Protection

Objectives

- Describe the general organization of a computer system and the role of interrupts
- Describe the components in a modern, multiprocessor computer system
- Illustrate the transition from user mode to kernel mode
- Discuss how operating systems are used in various computing environments

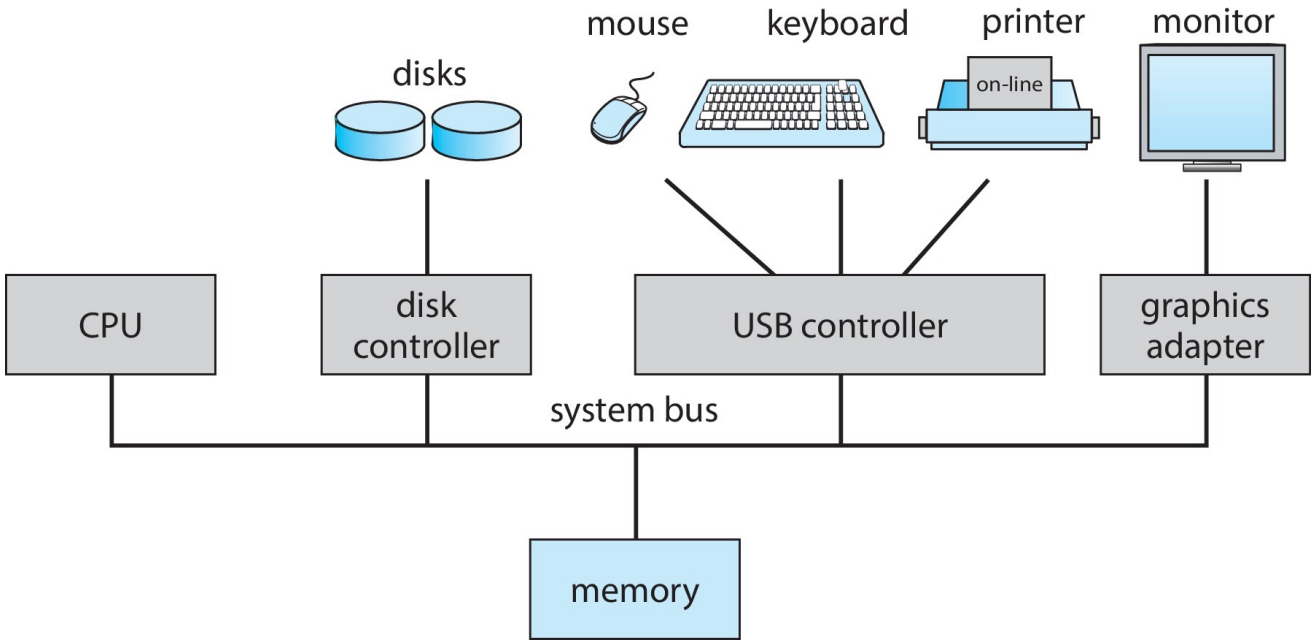
โครงสร้างของระบบคอมพิวเตอร์

- ส่วนประกอบของระบบคอมพิวเตอร์
- การประสานงานระหว่างส่วนประกอบของระบบคอมพิวเตอร์ด้วย Interrupts
- การเก็บข้อมูลในระบบคอมพิวเตอร์
- การประมวลผลแบบ User Mode และ Kernel Mode

Overview of Computer System Structure

Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through common **bus** providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles



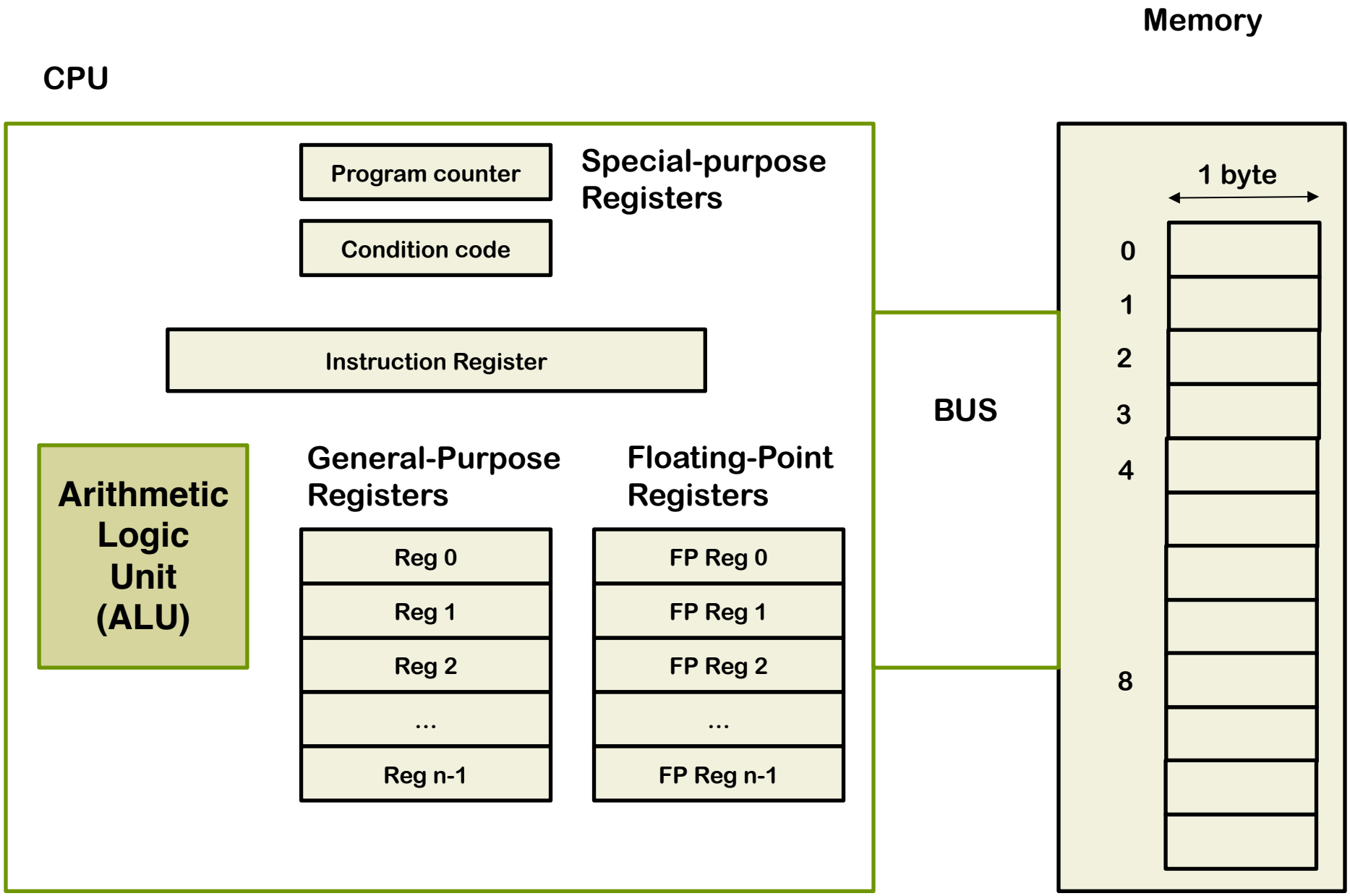
การประมวลผลของซีพียู

- การสั่งงานระบบคอมพิวเตอร์จะเกิดจากการสั่งงานผ่านซีพียูเท่านั้น
  - โปรแกรมไม่สามารถสั่งงานอุปกรณ์ภายในได้โดยตรง
  - โปรแกรมสั่งงานด้วยชุดคำสั่งเรียกว่า Instruction Set Architecture (ISA)
- CPU ทำหน้าที่อ่านคำสั่งและข้อมูลจากหน่วยความจำมาประมวลผลและส่งผลลัพธ์ไปเก็บในหน่วยความจำ
- หน่วยความจำหลักหรือ Main Memory ใช้เก็บ
  - (1) ข้อมูลชุดคำสั่ง
  - (2) ข้อมูลสำหรับการประมวลผลและ
  - (3) ผลลัพธ์ของการประมวลผล
- ซีพียูนำข้อมูลจากหน่วยความจำเข้ามาประมวลผลทีละคำสั่ง เมื่อเสร็จแล้วจึงประมวลผลคำสั่งถัดไป
- คำถามท้าย Lecture: Machine Cycle คืออะไร

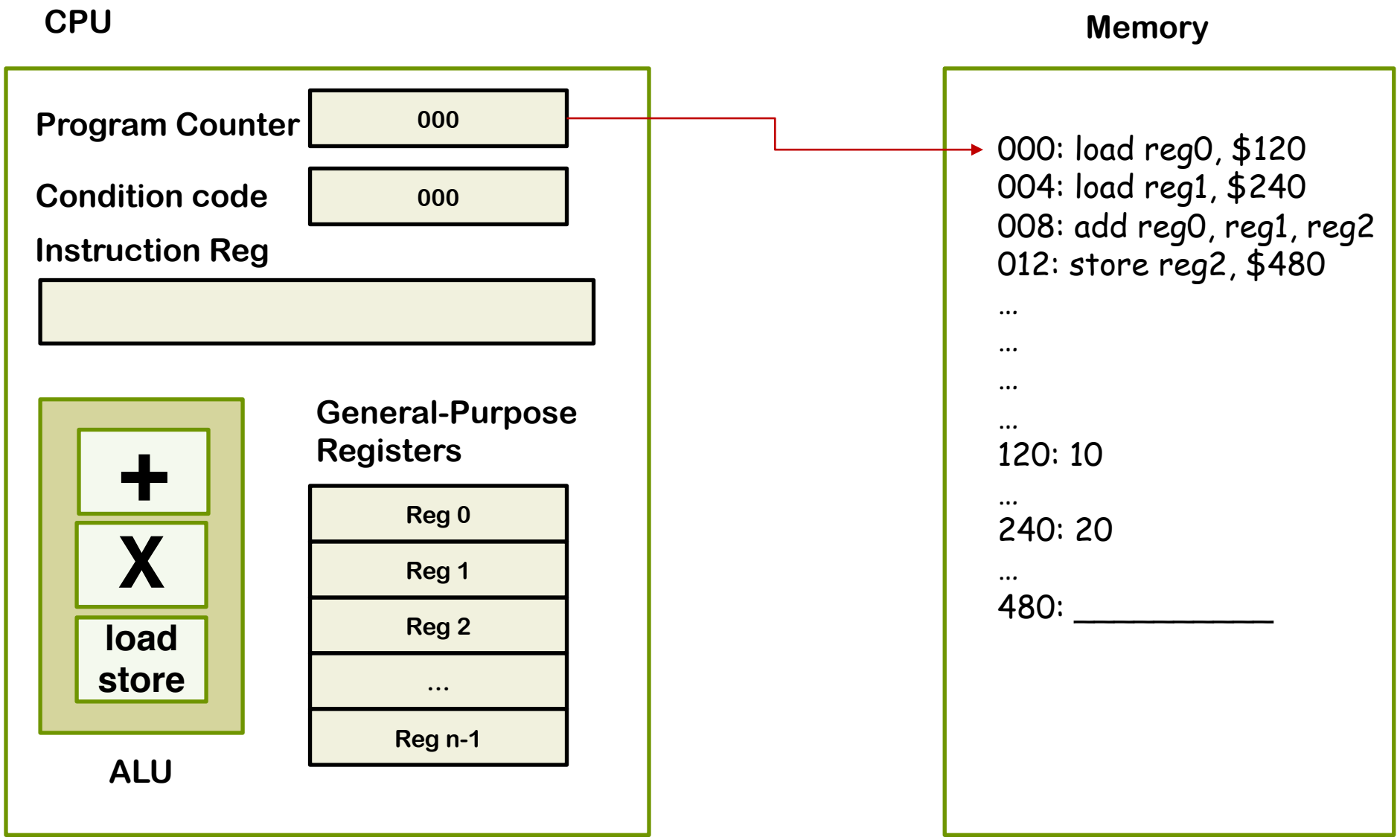
การประมวลผลชุดคำสั่งของซีพียู

- CPU จะนำคำสั่งและข้อมูลมาจาก **Memory** เพื่อประมวลผลและนำผลลัพธ์ไปเก็บไว้ใน **Memory** และเรียกคำสั่งถัดไปมาประมวลผลต่อไป
- ใน CPU จะมีพื้นที่รับข้อมูล Input (จาก Memory) ที่จะนำไปประมวลผลและเก็บข้อมูล Output ที่อาจถูกส่งต่อไปเก็บใน Memory เรียกว่า **Registers**
- ใน CPU จะมี Registers พิเศษ ที่ระบุสถานะของ hardware อันเกิดจากผลของการประมวลผลคำสั่ง เรียกว่า **Condition Code registers**
- แสดงตัวอย่างการประมวลผลของซีพียูในสไลด์ถัดไป
- คำถามท้าย Lecture: ขอให้ นศ จำลองต่อให้จบ และแสดงค่าในหน่วยความจำ

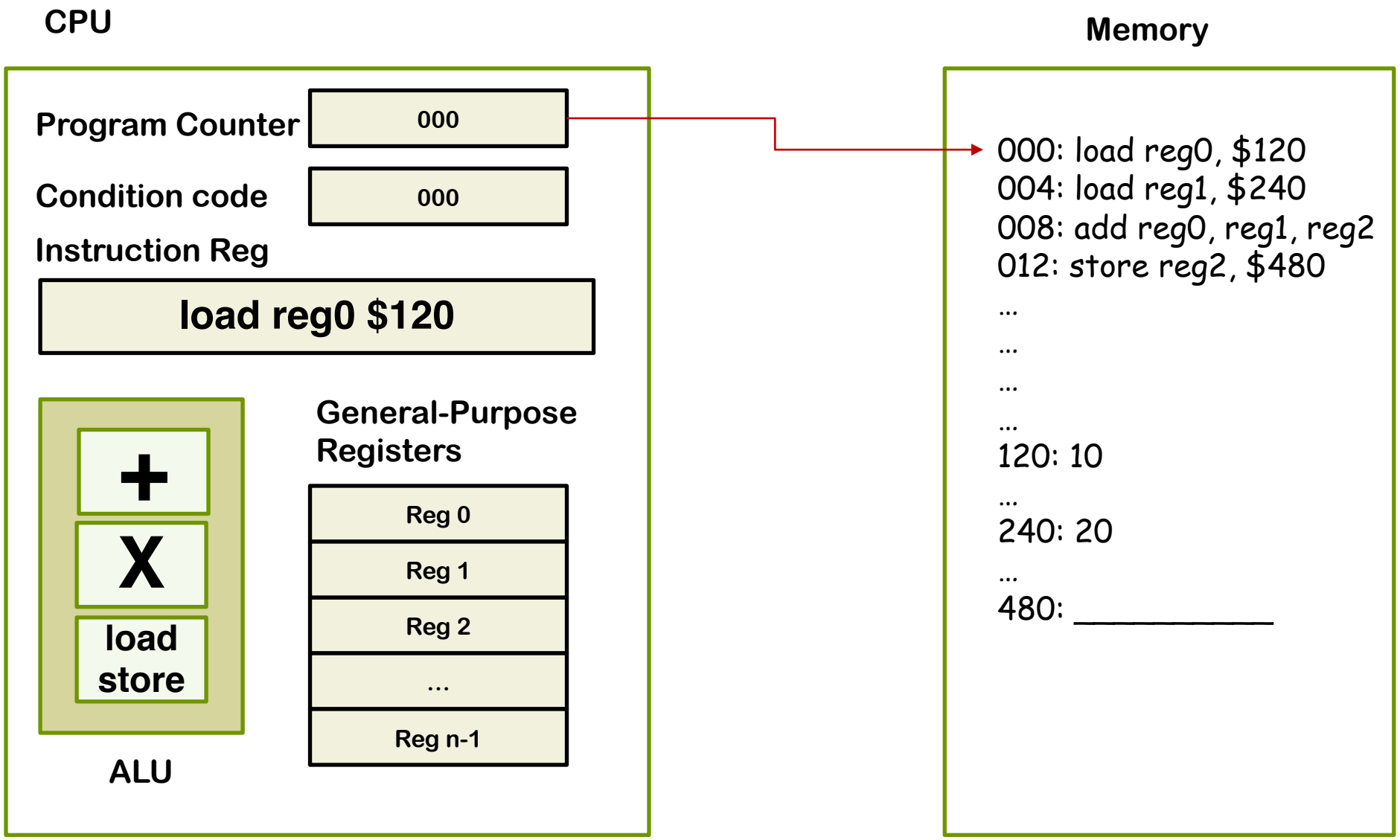
ซีพียูและหน่วยความจำ



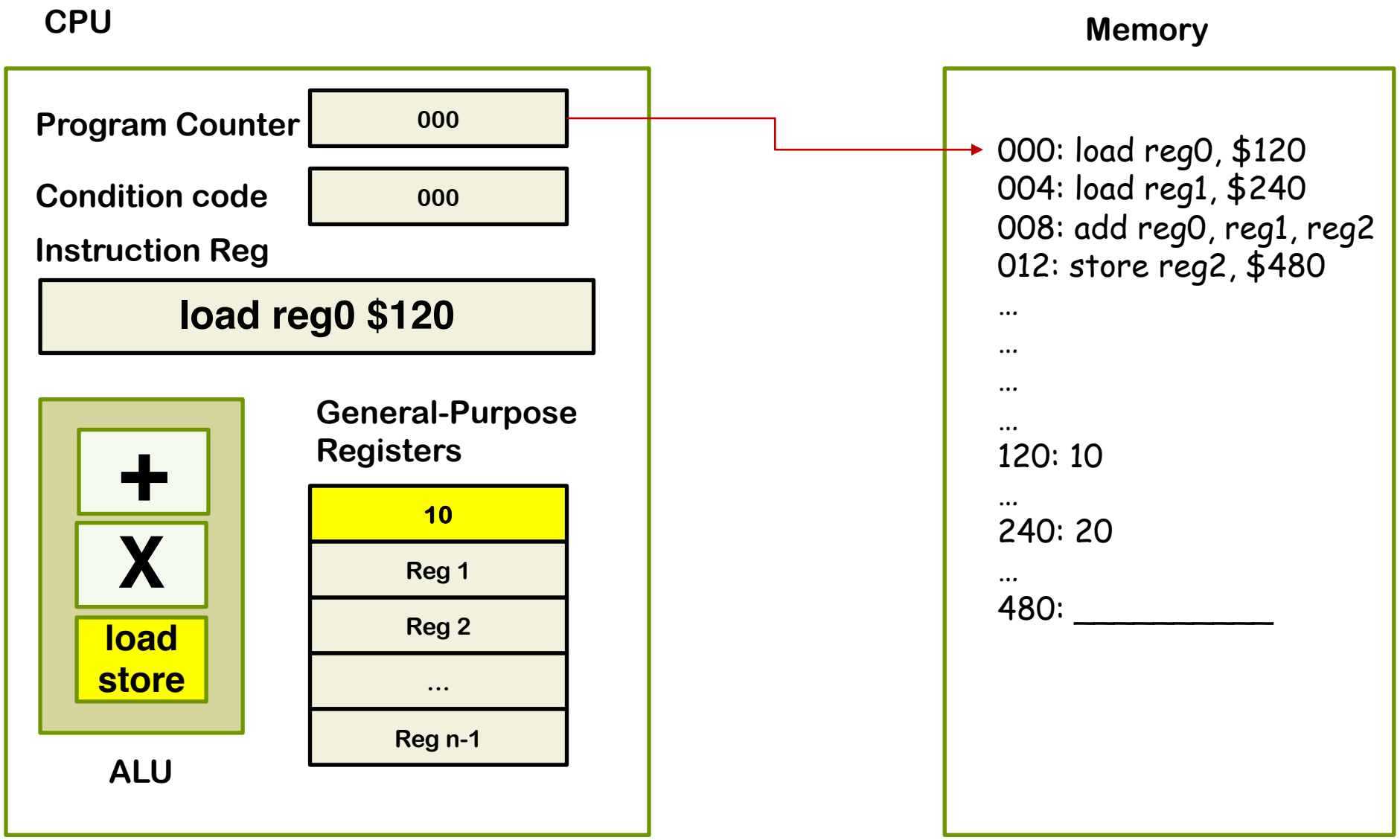
ตัวอย่างการประมวลผลของซีพียู (32 บิต)



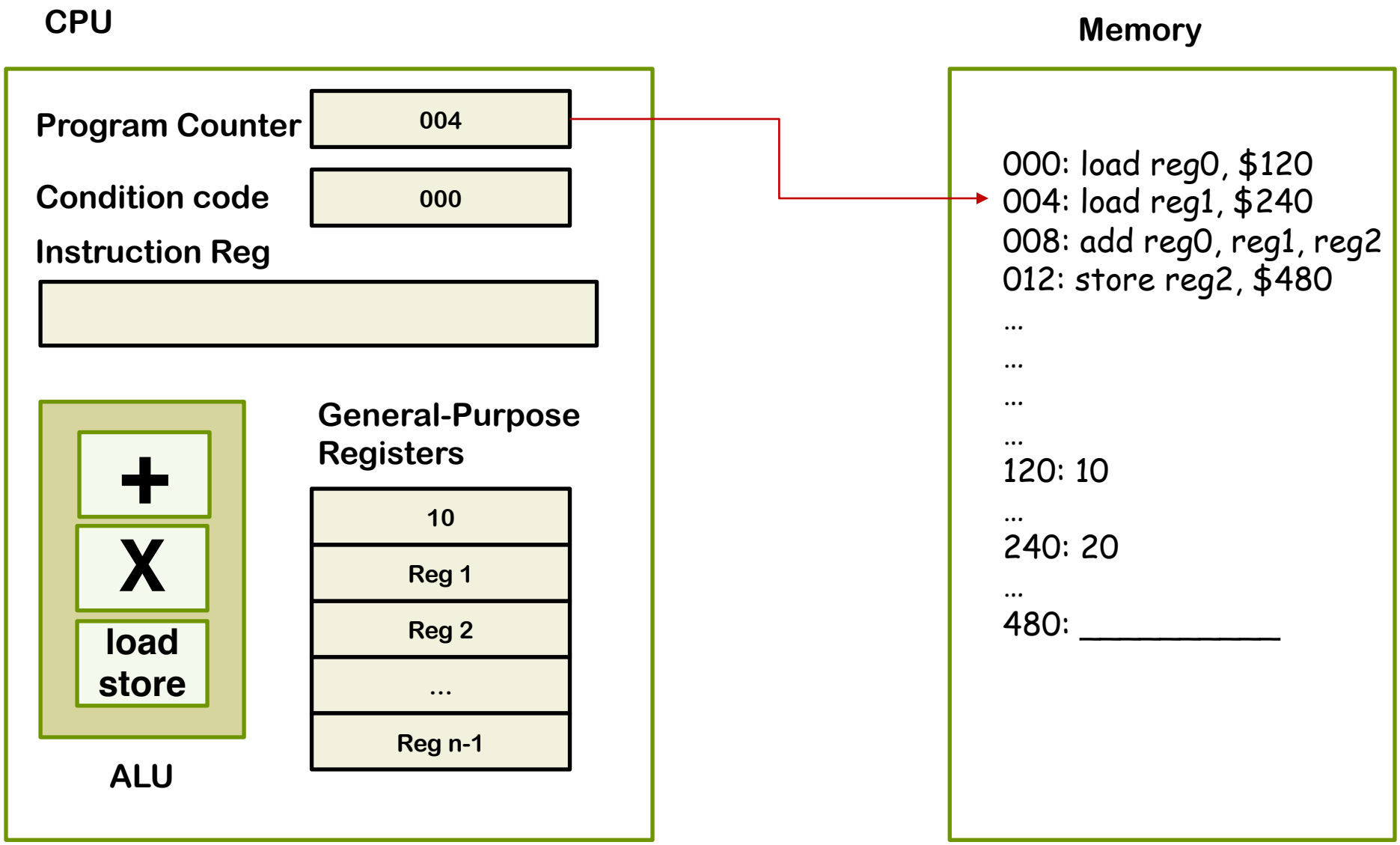
CPU at work


CPU
Memory
Program Counter
000
Condition code
000
Instruction Reg
load reg0 $120
+
X
load
store
ALU
General-Purpose
Registers
Reg 0
Reg 1
Reg 2
...
Reg n-1
000: load reg0, $120
004: load reg1, $240
008: add reg0, reg1, reg2
012: store reg2, $480
...
...
...
...
120: 10
...
240: 20
...
480: _____

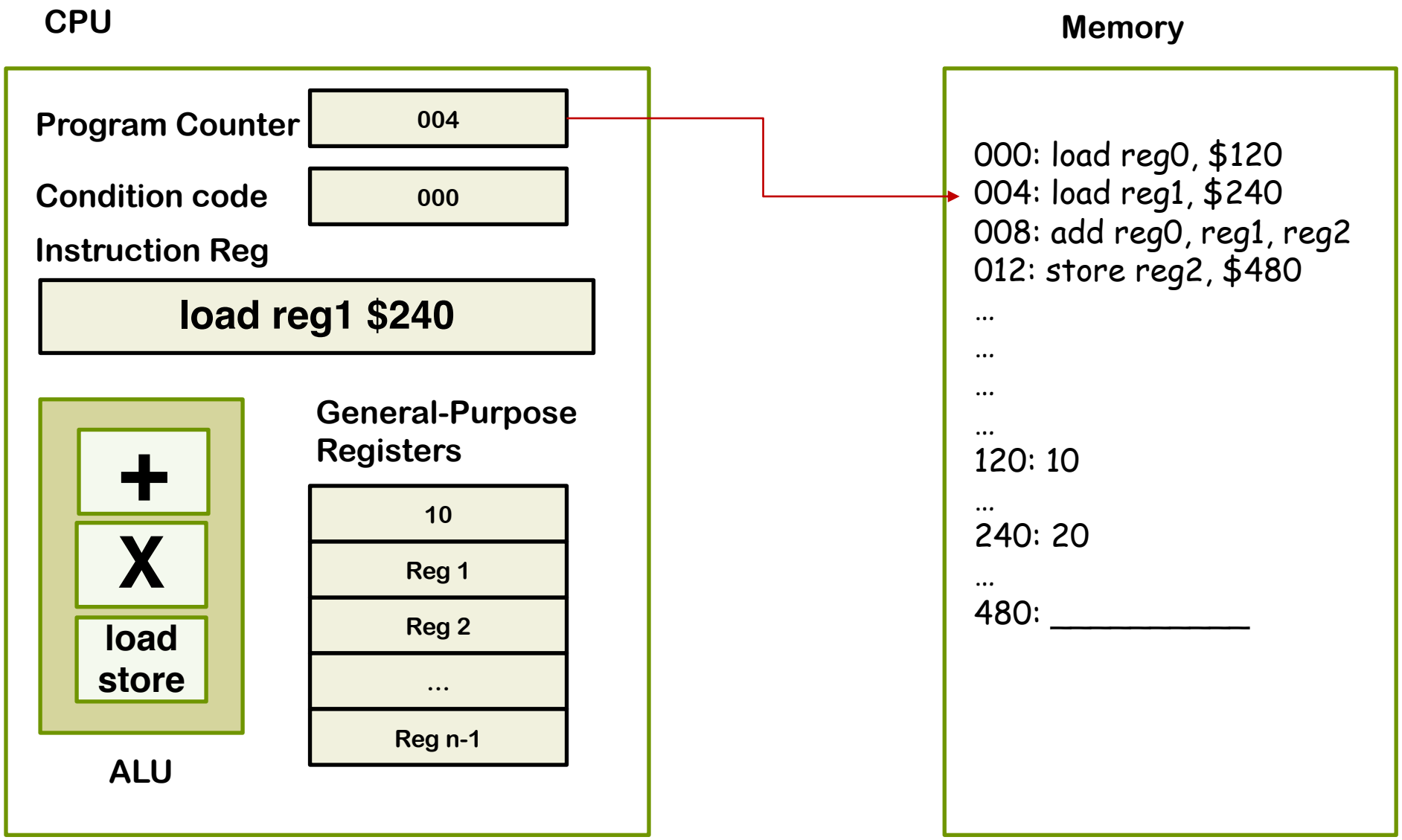
CPU at work


CPU
Memory
Program Counter
000
Condition code
000
Instruction Reg
load reg0 $120
+
X
load
store
ALU
General-Purpose
Registers
10
Reg 1
Reg 2
…
Reg n-1
000: load reg0, $120
004: load reg1, $240
008: add reg0, reg1, reg2
012: store reg2, $480
…
…
…
…
120: 10
…
240: 20
…
480: _____

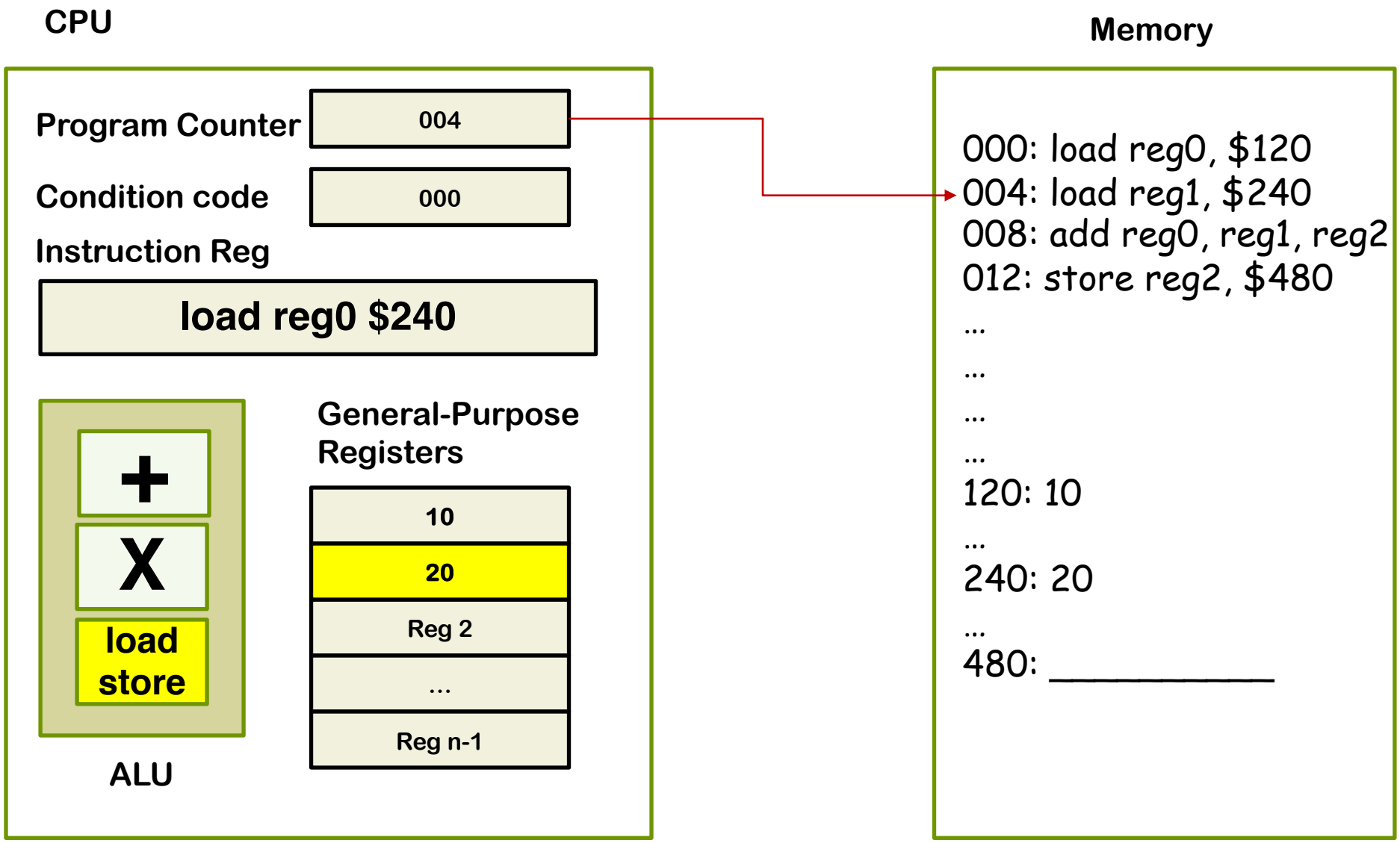
CPU at work


CPU
Memory
Program Counter
004
Condition code
000
Instruction Reg
+
X
load
store
ALU
General-Purpose
Registers
10
Reg 1
Reg 2
...
Reg n-1
000: load reg0, $120
004: load reg1, $240
008: add reg0, reg1, reg2
012: store reg2, $480
...
...
...
...
120: 10
...
240: 20
...
480: _____

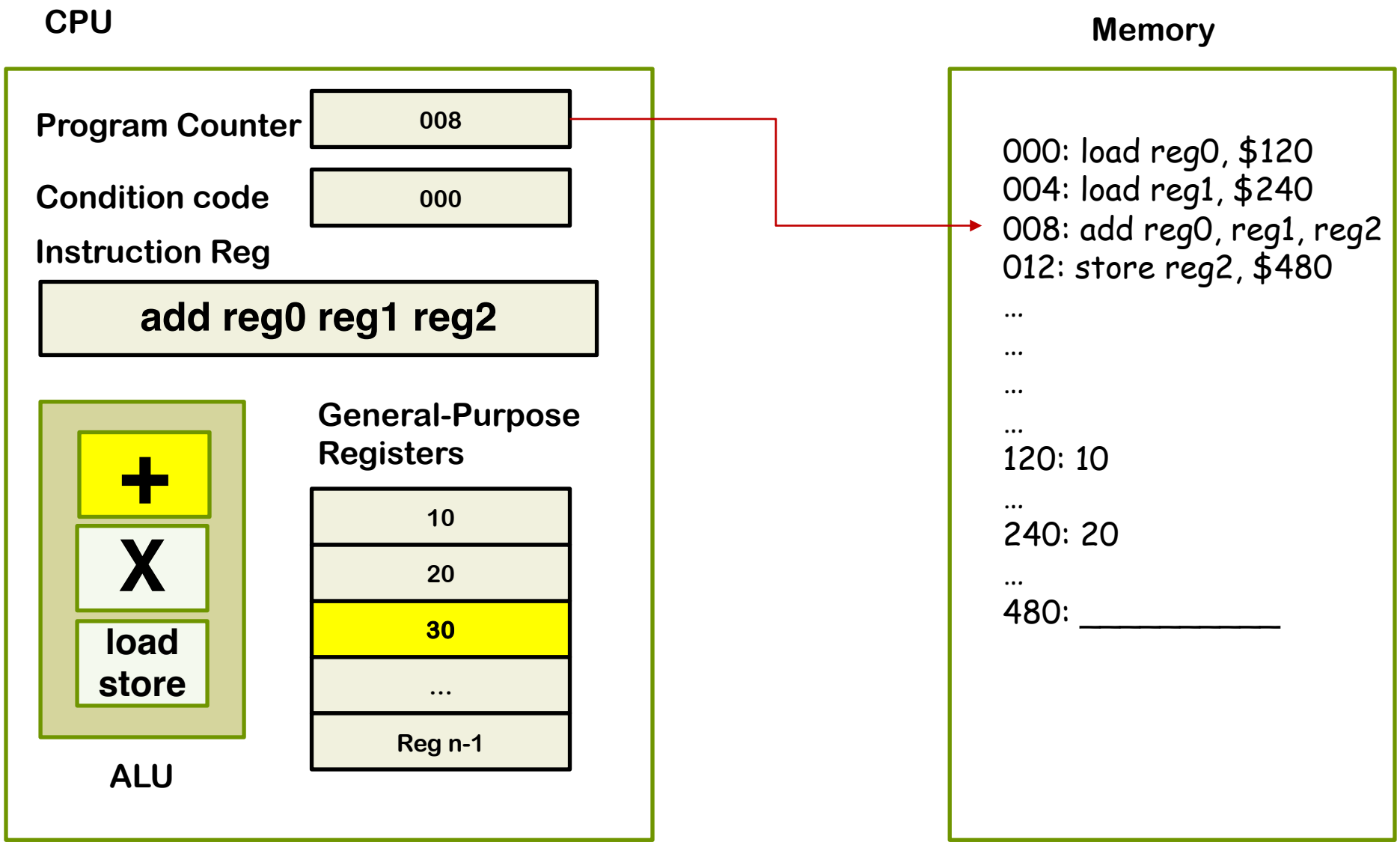
CPU at work


CPU
Memory
Program Counter
004
Condition code
000
Instruction Reg
load reg1 $240
+
X
load
store
ALU
General-Purpose
Registers
10
Reg 1
Reg 2
...
Reg n-1
000: load reg0, $120
004: load reg1, $240
008: add reg0, reg1, reg2
012: store reg2, $480
...
...
...
...
120: 10
...
240: 20
...
480: _____

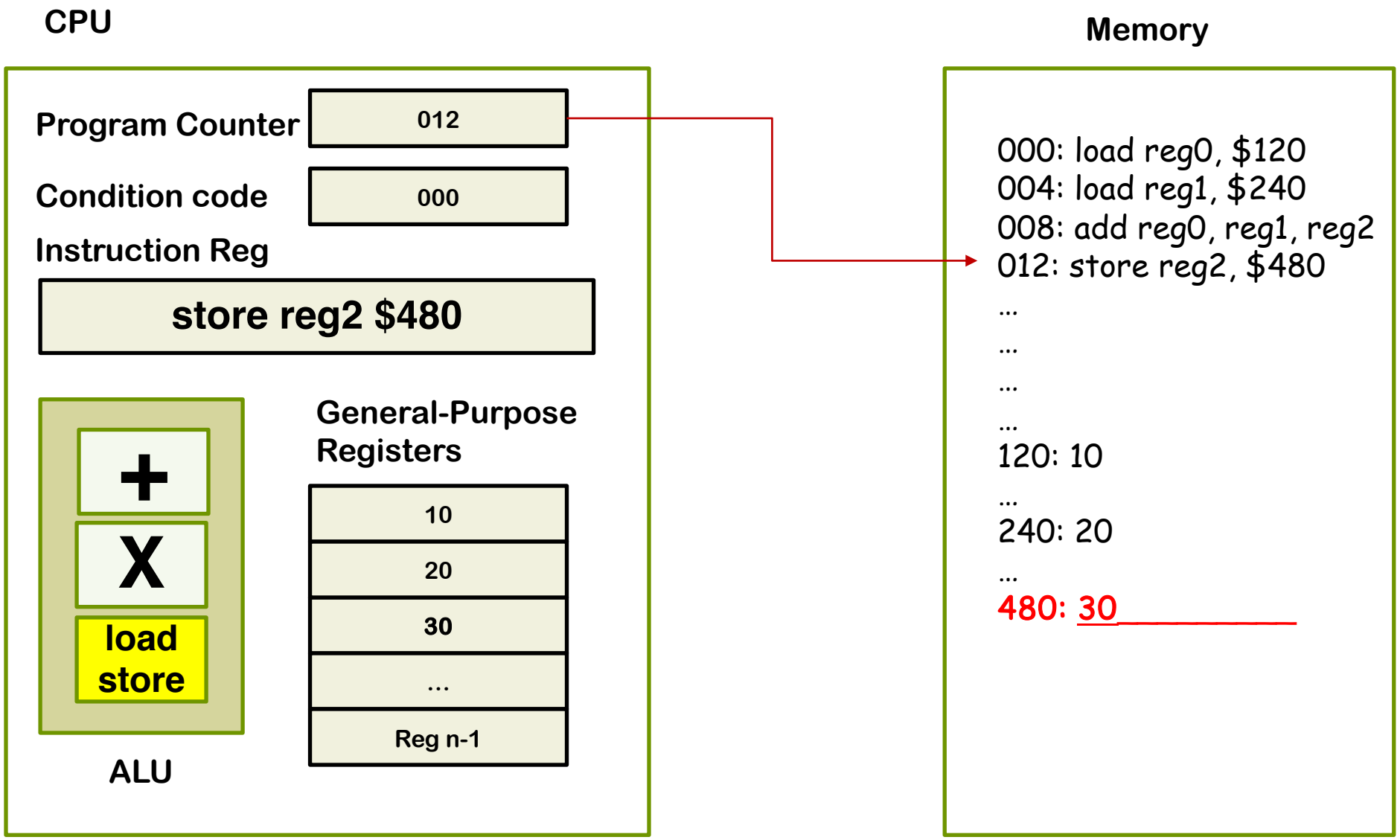
CPU at work


CPU
Memory
Program Counter
004
Condition code
000
Instruction Reg
load reg0 $240
+
X
load
store
ALU
General-Purpose
Registers
10
20
Reg 2
...
Reg n-1
000: load reg0, $120
004: load reg1, $240
008: add reg0, reg1, reg2
012: store reg2, $480
...
...
...
...
120: 10
...
240: 20
...
480: _____

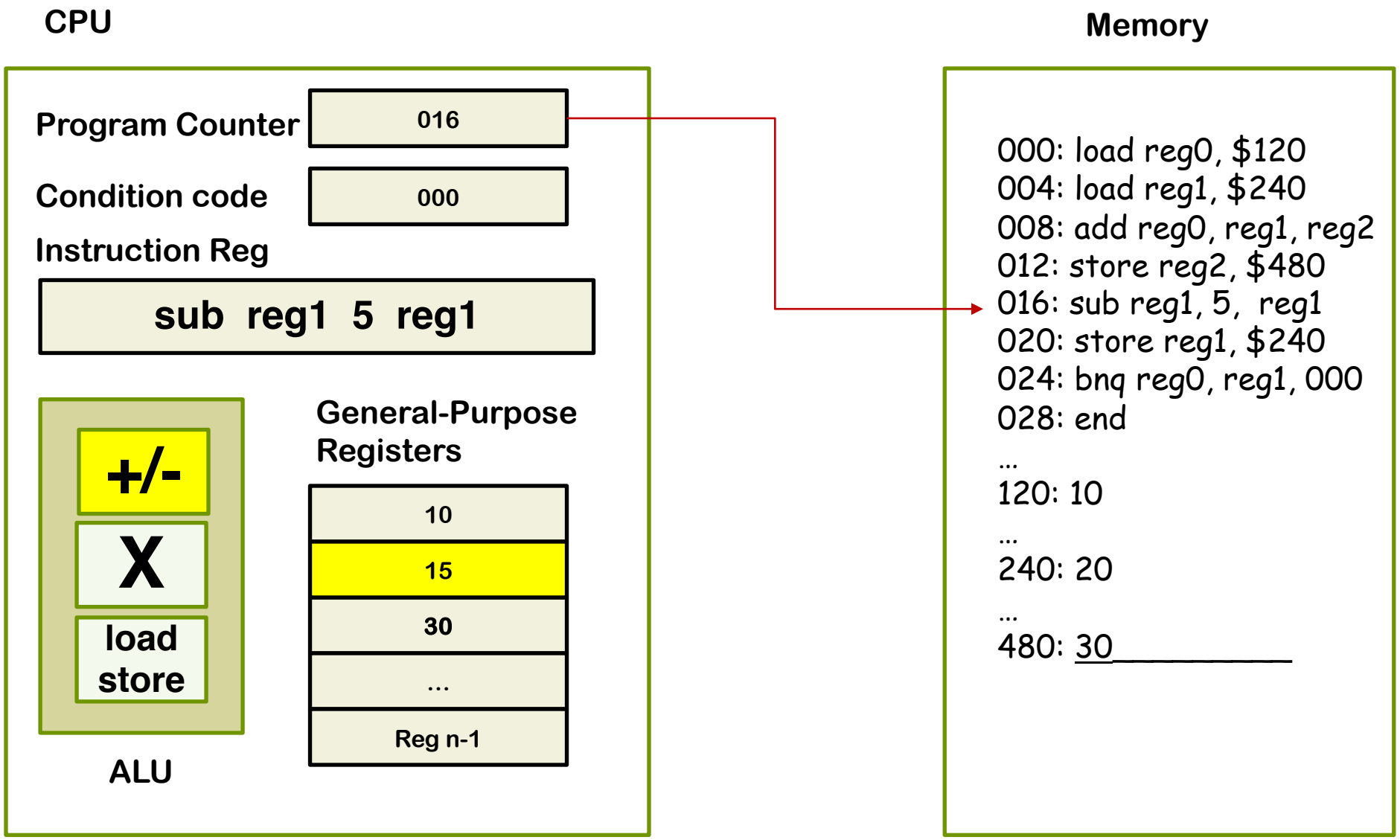
CPU at work


CPU
Memory
Program Counter
008
Condition code
000
Instruction Reg
add reg0 reg1 reg2
+
X
load
store
ALU
General-Purpose
Registers
10
20
30
...
Reg n-1
000: load reg0, $120
004: load reg1, $240
008: add reg0, reg1, reg2
012: store reg2, $480
...
...
...
...
120: 10
...
240: 20
...
480: _____

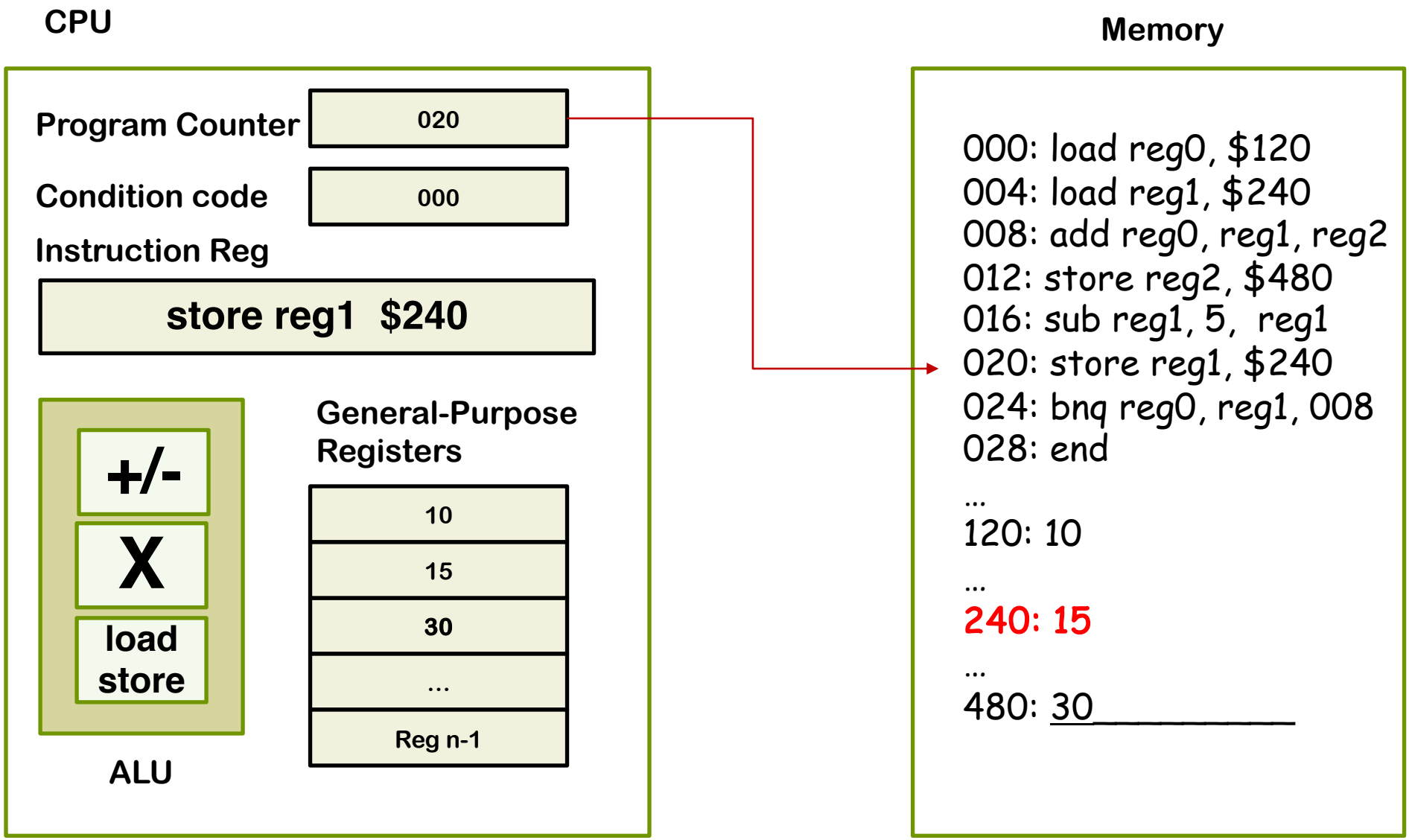
CPU at work


CPU
Memory
Program Counter
012
Condition code
000
Instruction Reg
store reg2 $480
+
X
load
store
ALU
General-Purpose
Registers
10
20
30
...
Reg n-1
000: load reg0, $120
004: load reg1, $240
008: add reg0, reg1, reg2
012: store reg2, $480
...
...
...
...
120: 10
...
240: 20
...
480: 30

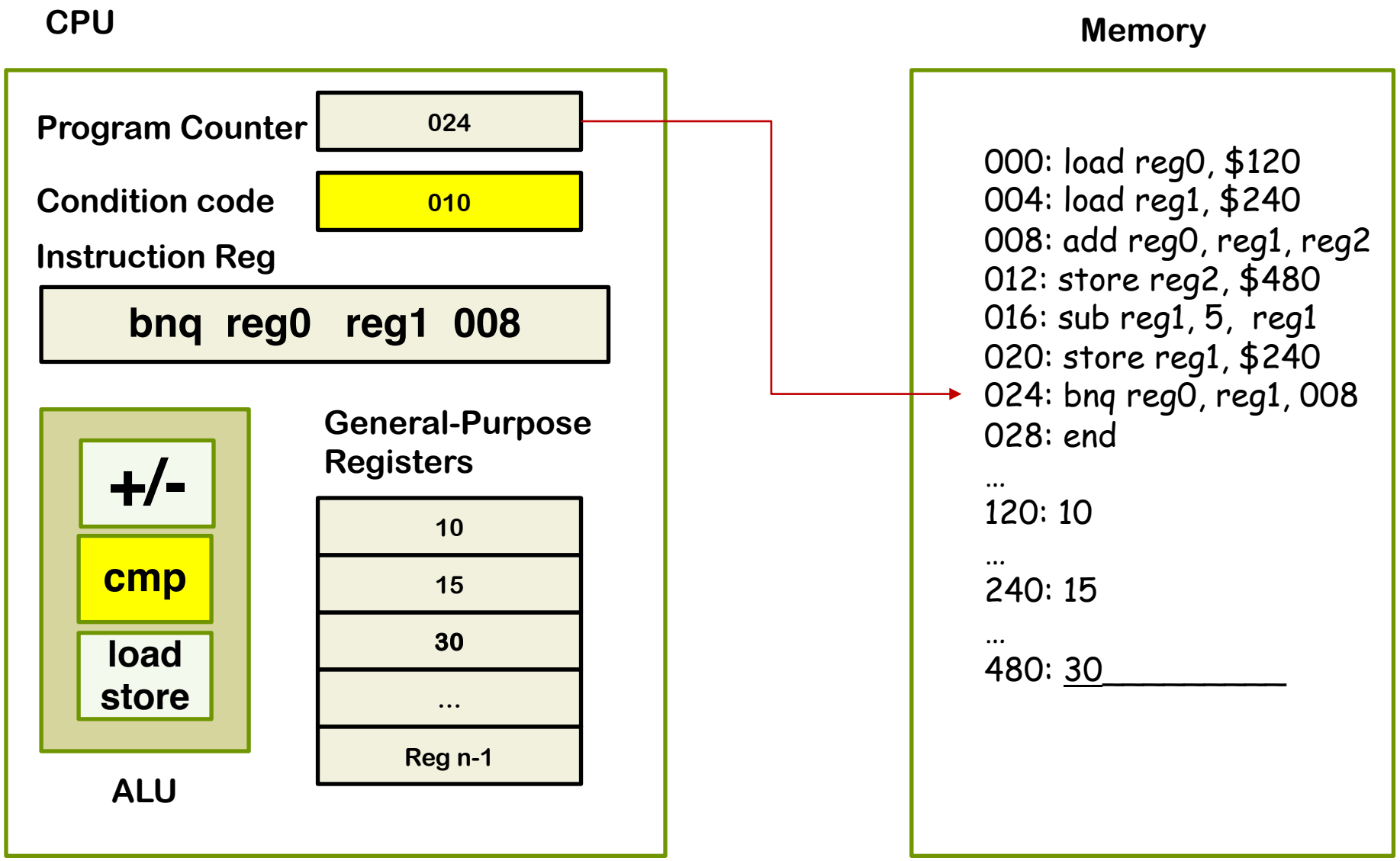
CPU at work


CPU
Memory
Program Counter
016
Condition code
000
Instruction Reg
sub reg1 5 reg1
+/-
X
load
store
ALU
General-Purpose
Registers
10
15
30
...
Reg n-1
000: load reg0, $120
004: load reg1, $240
008: add reg0, reg1, reg2
012: store reg2, $480
016: sub reg1, 5, reg1
020: store reg1, $240
024: bnq reg0, reg1, 000
028: end
...
120: 10
...
240: 20
...
480: 30

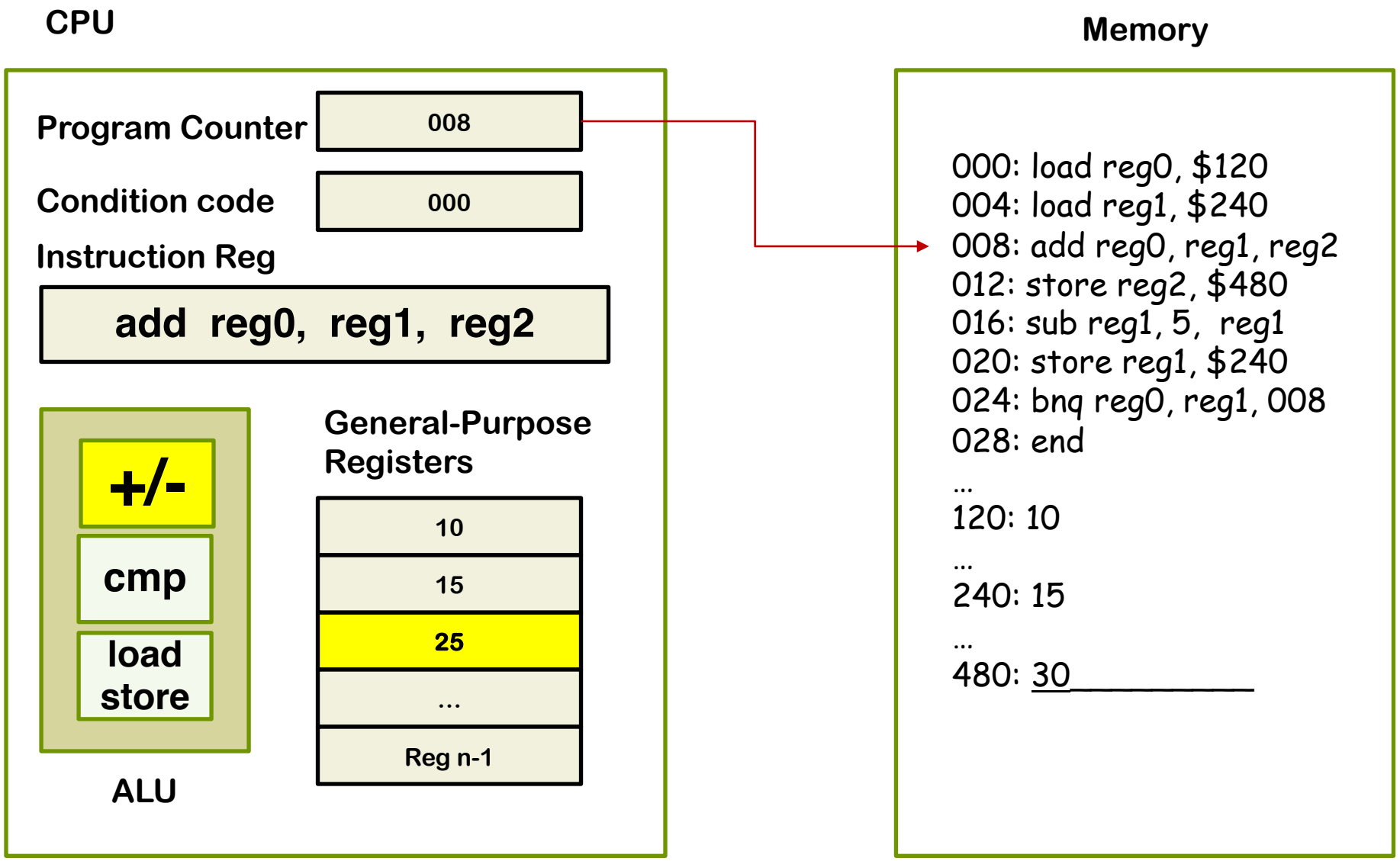
CPU at work


CPU
Program Counter
020
Condition code
000
Instruction Reg
store reg1 $240
+/-
X
load
store
ALU
General-Purpose
Registers
10
15
30
...
Reg n-1
Memory
000: load reg0, $120
004: load reg1, $240
008: add reg0, reg1, reg2
012: store reg2, $480
016: sub reg1, 5, reg1
020: store reg1, $240
024: bnq reg0, reg1, 008
028: end
...
120: 10
...
240: 15
...
480: 30

CPU at work


CPU
Program Counter
024
Condition code
010
Instruction Reg
bnq reg0 reg1 008
+/-
cmp
load
store
ALU
General-Purpose
Registers
10
15
30
...
Reg n-1
Memory
000: load reg0, $120
004: load reg1, $240
008: add reg0, reg1, reg2
012: store reg2, $480
016: sub reg1, 5, reg1
020: store reg1, $240
024: bnq reg0, reg1, 008
028: end
...
120: 10
...
240: 15
...
480: 30

CPU at work


CPU
Memory
Program Counter
008
Condition code
000
Instruction Reg
add reg0, reg1, reg2
+/-
cmp
load
store
ALU
General-Purpose
Registers
10
15
25
...
Reg n-1
000: load reg0, $120
004: load reg1, $240
008: add reg0, reg1, reg2
012: store reg2, $480
016: sub reg1, 5, reg1
020: store reg1, $240
024: bnq reg0, reg1, 008
028: end
...
120: 10
...
240: 15
...
480: 30

การขัดจังหวะในการทำงาน

- ในขณะที่เราทำงาน เราสามารถถูกขัดจังหวะได้เสมอ เมื่อมีเหตุการณ์เกิดขึ้น
- ตย. ผู้จัดการร้านสะดวกซื้อกำลังเช็ค stock สินค้าในร้าน สามารถถูกขัดจังหวะโดยลูกค้าที่เข้ามาสอบถามหรือซื้อของได้ หรือเมื่อมีเหตุการณ์เร่งด่วน และผู้จัดการต้องเปลี่ยนการทำงานไปจัดการเหตุการณ์เร่งด่วนนั้น และเมื่อเสร็จแล้วก็กลับมาเช็ค stock สินค้าในร้านต่อ
- การทำงานของคนเราก็เช่นกัน ระหว่างทำงานสามารถมีเหตุการณ์เกิดขึ้น
- เราต้องเปลี่ยนการทำงานไปจัดการเหตุการณ์นั้น
- แล้วค่อยกลับมาทำงานเดิมต่อ
- เหตุการณ์อาจมาจากภายในตัวเรา เช่น หิวข้าว หรือจากภายนอก เช่น มีเพื่อนมาคุยด้วย

การขัดจังหวะในการทำงาน

- เหตุการณ์ที่เกิดขึ้นอาจมีหลายเหตุการณ์เกิดขึ้นพร้อมกัน
- เหตุการณ์มีหลายชนิด
- เหตุการณ์ที่หลีกเลี่ยงไม่ได้ต้องจัดการทันที: ไฟไหม้ เจ็บป่วย
- เหตุการณ์ที่สามารถหลีกเลี่ยงและจัดการได้:
  - เราจัดการเหตุการณ์ที่สำคัญก่อนและเลื่อนเหตุการณ์ที่สำคัญน้อยกว่าไปจัดการที่หลังในเวลาที่เหมาะสม
  - เราอาจเลือกที่จะไม่สนใจเหตุการณ์บางอย่างได้

Instruction Cycle With Interrupts

- การขัดจังหวะใน CPU เรียกว่า Interrupts
- CPU ได้รับการออกแบบให้เช็คสัญญาณวงจรหนึ่งที่เชื่อมต่อกับ CPU ว่ามี Interrupt เกิดขึ้นหรือไม่ก่อนที่จะ fetch คำสั่งใหม่ของ Process ที่ใช้งาน CPU อยู่ในปัจจุบัน
- ถ้ามี Interrupt ก็ให้ CPU ไป fetch คำสั่งของ Interrupt Handler มา Execute เป็นคำสั่งถัดไป
- แต่ถ้าไม่มีก็ให้ fetch คำสั่งถัดไปของ Process เข้ามา Execute
- ดังภาพถัดไป

Instruction Cycle with Interrupts

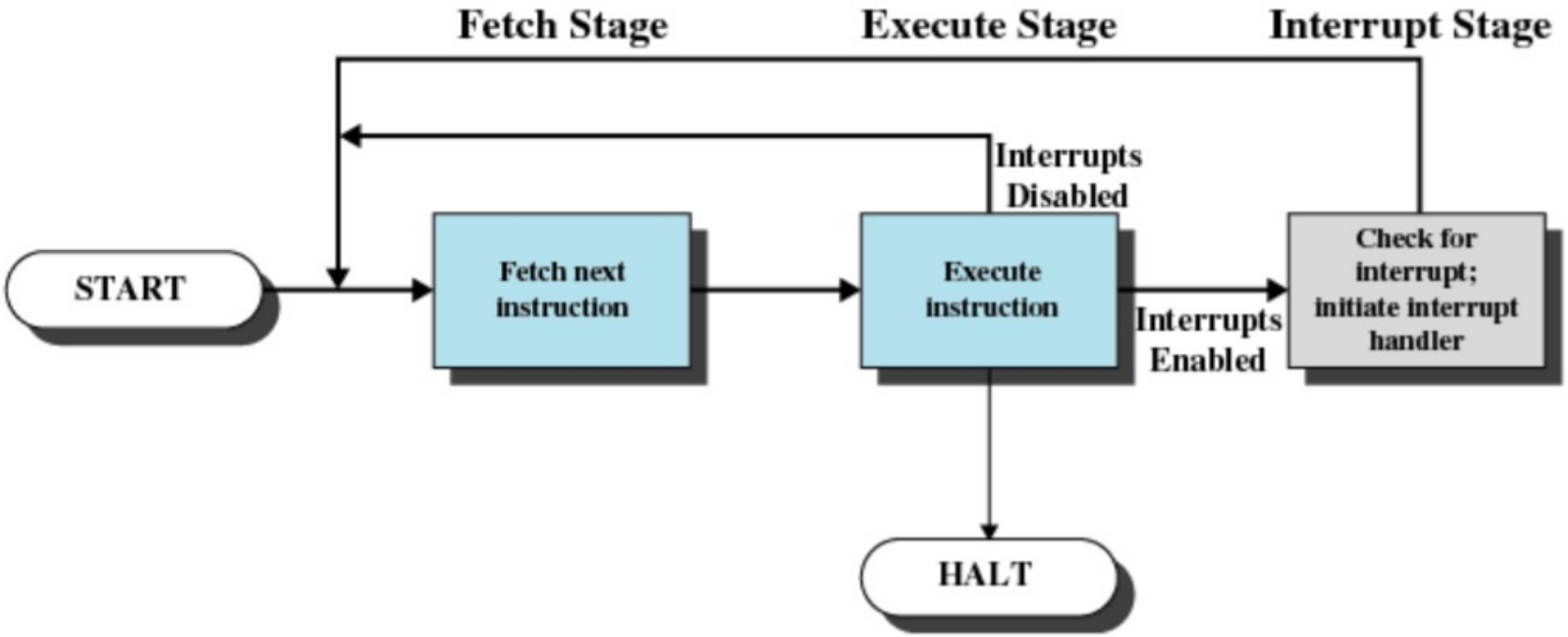


Figure 1.7 Instruction Cycle with Interrupts

Reference: William Stalling, Operating Systems: Internal and Design Principles

องค์ประกอบในระบบคอมพิวเตอร์

- ซีพียูประมวลผลโดยทำตามชุดคำสั่งของโปรแกรมในหน่วยความจำ
- เมื่อโปรแกรมต้องการติดต่ออุปกรณ์ ไอโอ ซีพียูก็จะรันคำสั่งของ OS เรียกว่า Device driver ที่ทำหน้าที่จัดการอุปกรณ์ไอโอที่เกี่ยวข้อง
- Device driver จะรันคำสั่งที่ส่งผลให้ซีพียูส่งสัญญาณไปสั่งงานให้วงจร ไอโอ คอนโทรเลอร์ (I/O Controller) ประมวลผลให้
- อุปกรณ์ไอโอในระบบคอมพิวเตอร์มีหลาย อุปกรณ์
- ในระหว่างที่ซีพียูประมวลผลคำสั่ง ก็จะต้องใช้งานอุปกรณ์เหล่านั้น ซึ่งแต่ละอุปกรณ์มีวงจร Controller ควบคุม
- วงจร Controller เป็น Special Purpose processor ที่มีสถานการณ์ทำงานภายในที่บางส่วนซีพียูไม่สามารถเข้าถึงได้ (และไม่จำเป็นต้องรู้)
  - เหมือนการทำงานภายในองค์กร ซีอีโอ หรือหัวหน้าฝ่ายไม่จำเป็นต้องรู้เรื่องปลีกย่อย

องค์ประกอบในระบบคอมพิวเตอร์

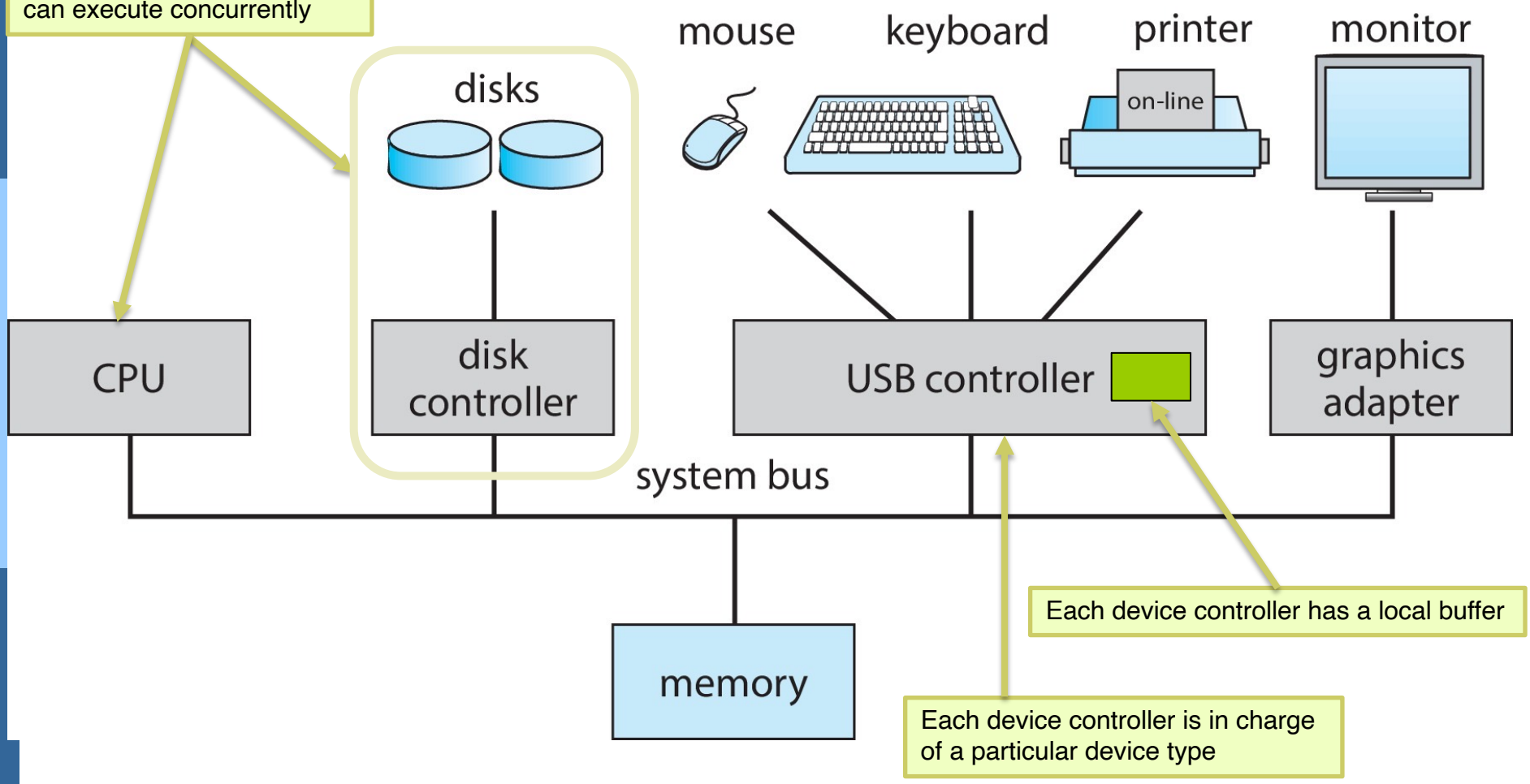
- ขอให้ นศ ลองนึกว่า เรา เป็นซีพียู และ เรามีผู้ช่วย คืออุปกรณ์ไอโอคอนโทรเลอร์ ซึ่งผู้ช่วยแต่ละคน มีความสามารถเฉพาะด้าน และจะทำงานตามคำสั่งของเรา
- เมื่อเราสั่งงานให้ ผู้ช่วยคนหนึ่งทำงานให้แล้ว เราสามารถไปทำงานอย่างอื่น ในระหว่างที่รอให้ผู้ช่วยคนนั้นทำงานเสร็จ
- เมื่อผู้ช่วยคนนั้นทำงานเสร็จ เขาก็จะตะโกนแจ้งให้เราทราบว่าเขาทำงานเสร็จแล้ว
- ในขณะนั้นเราอาจทำงานอื่นอยู่และเสียงตะโกนของเขาก็มาขัดจังหวะการทำงานของเรา ทำให้ เรา (ซีพียู) ต้องหยุดการทำงานที่ทำอยู่ชั่วคราวเพื่อไปจัดการเหตุการณ์ที่ผู้ช่วยคนนั้นแจ้งมา
- เราจัดการนำผลของการทำงานของผู้ช่วยไปใช้ประโยชน์
- เรากลับไปทำงานที่ทำค้างไว้ต่อ

Computer-System Operation

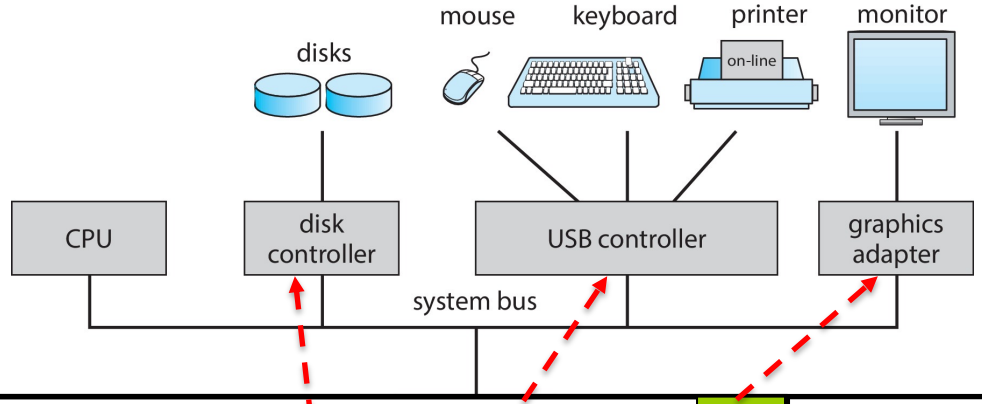
- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- Each device controller type has an operating system **device driver** to manage it
  - Device Driver คือชุดคำสั่งของโอเอส
  - ประกอบไปด้วยชุดคำสั่งของซีพียู สำหรับจัดการไอโอ
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller
- Device controller informs CPU that it has finished its operation by causing an **interrupt**

Devices and Their Local Buffers

I/O devices and the CPU can execute concurrently



Device Drivers



OS kernel

memory

คือชุดคำสั่งและข้อมูลของ OS ที่ฝังตัวอยู่ในหน่วยความจำตลอดเวลา

Device Drivers
- Disk
- Mouse,
- KB,
- PRN
- GPU

System Call Interfaces

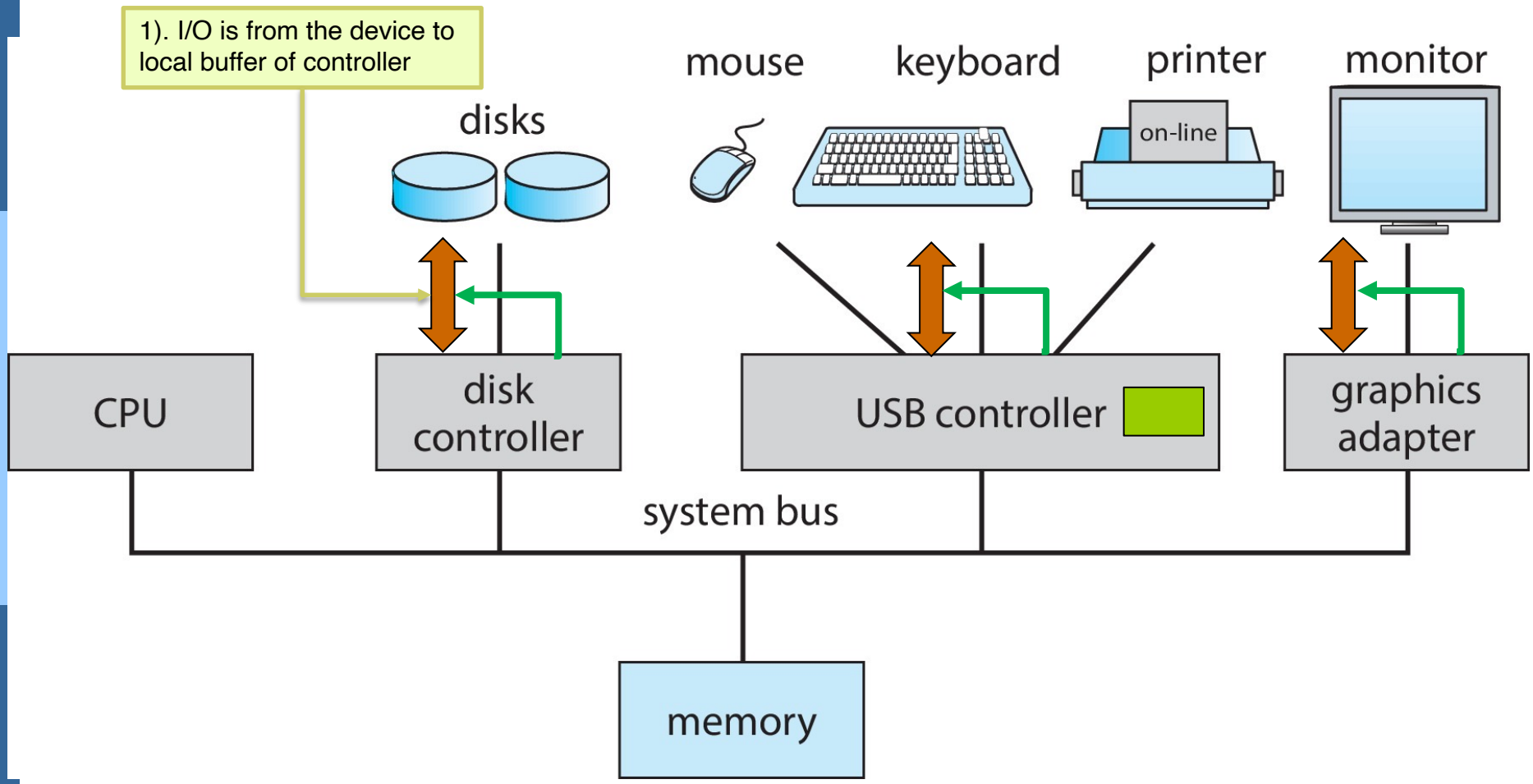
Application Program
…
Read file
SysCall()
…

Each device controller type has an operating system **device driver** to manage it

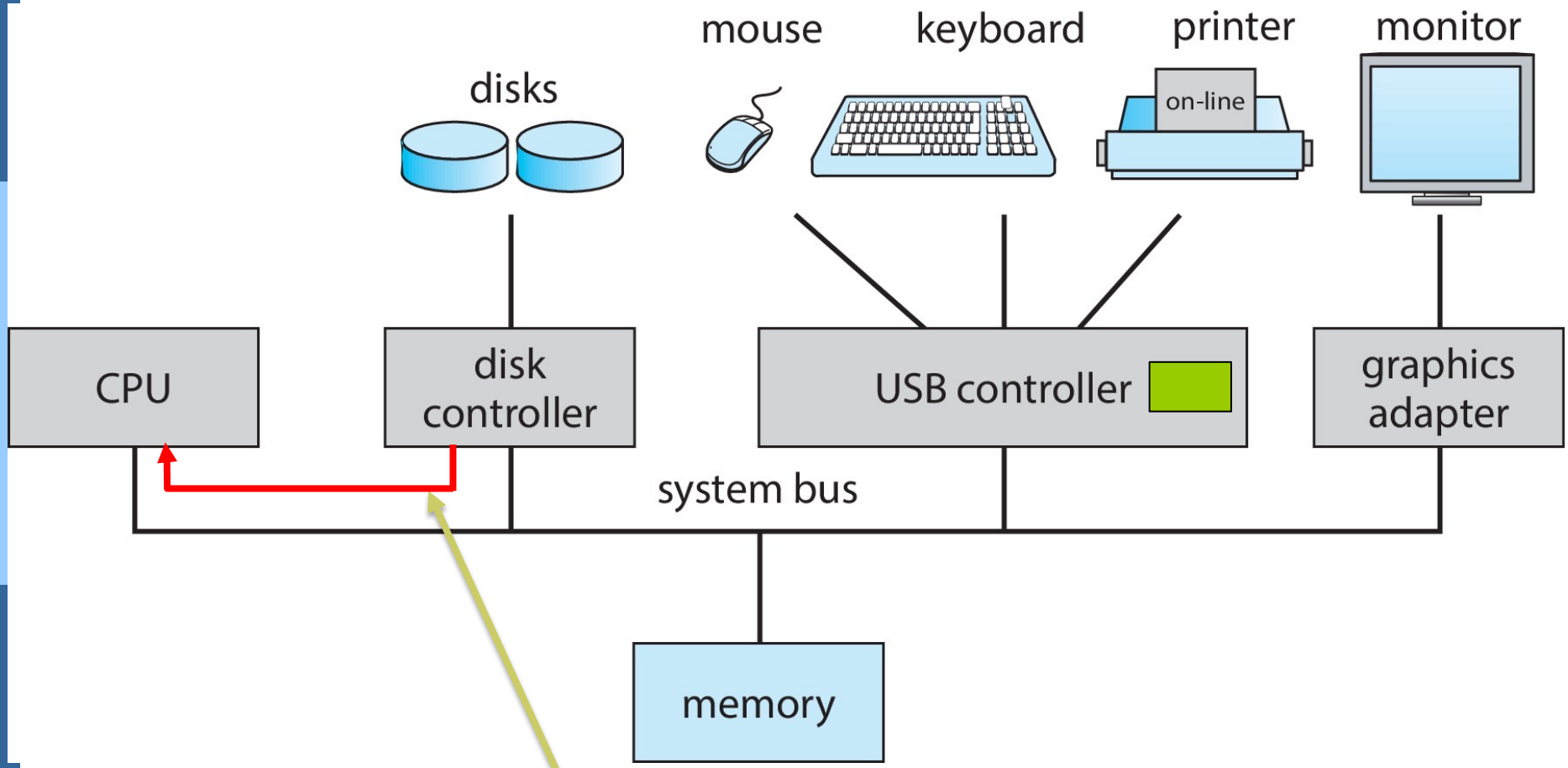
-Read file Routine Implementation
-Write file
-Read net
Etc.

คือชุดคำสั่งและข้อมูลของ App ที่ถูกโหลดจาก HDD/SSD มาในหน่วยความจำ เมื่อต้องการใช้งาน และลบออกไปจาก mem เมื่อจบการประมวลผล

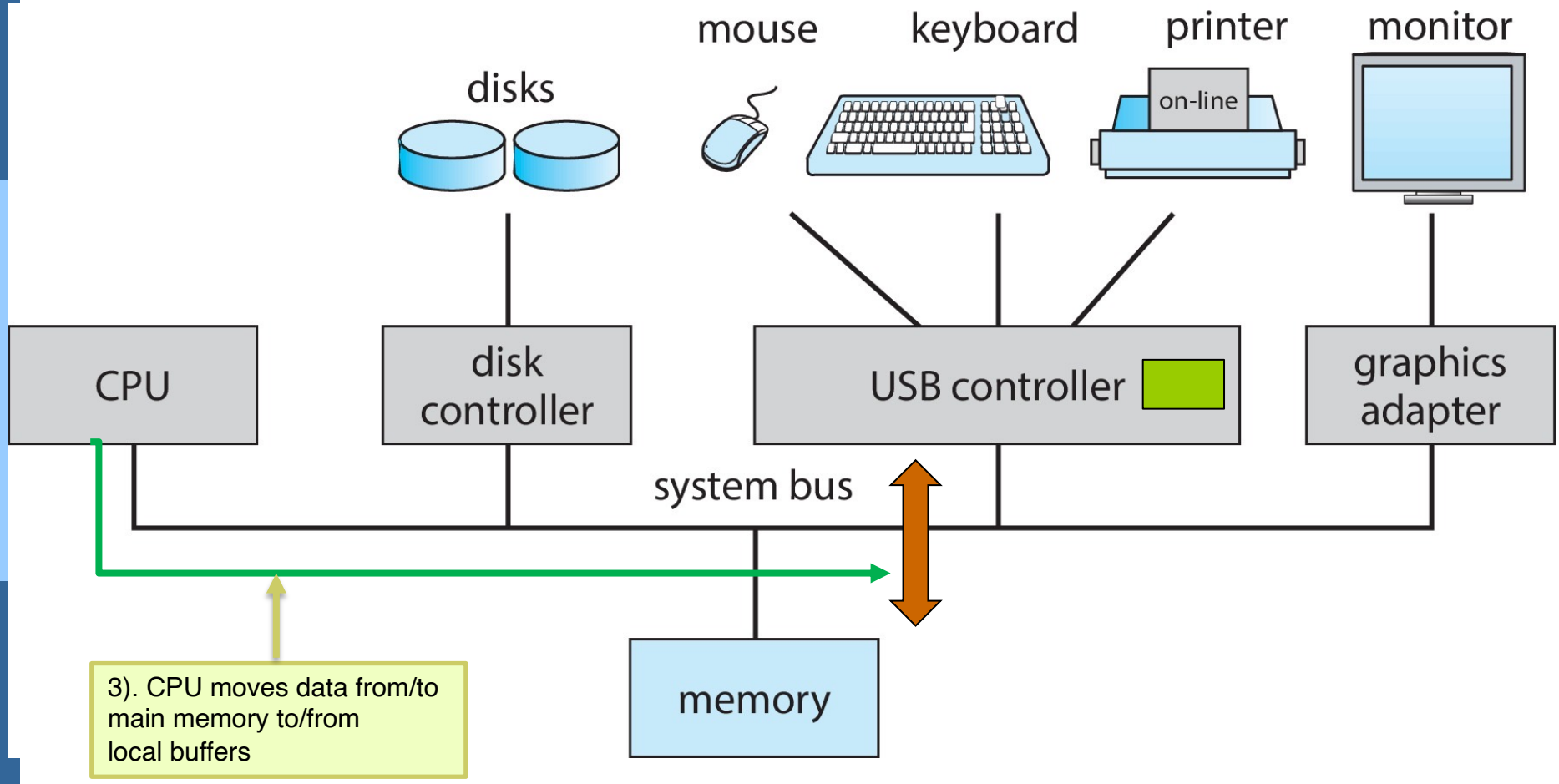
Computer System Operation (1)



Computer System Operation (2)

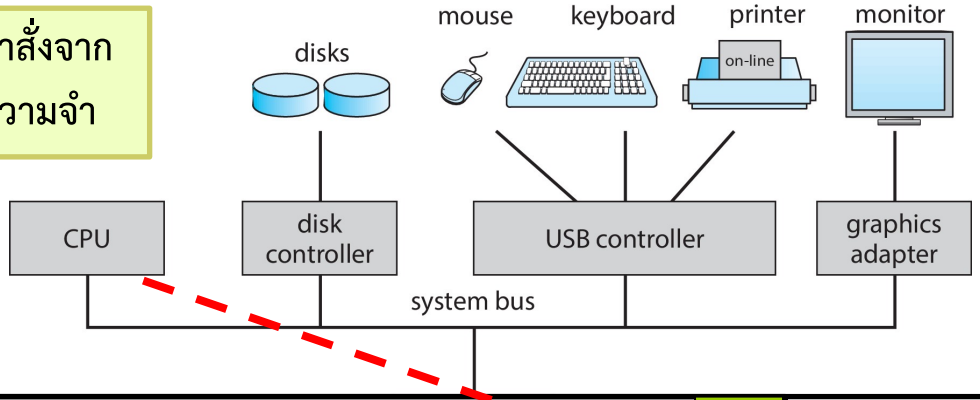


Computer System Operation (3)



E.g. 1. Read Data From a File

ซีพียู fetch และรันชุดคำสั่งจาก application ในหน่วยความจำ



memory

OS kernel

Device Drivers
- Disk
- Mouse,
- KB,
- PRN
- GPU

System Call Interfaces

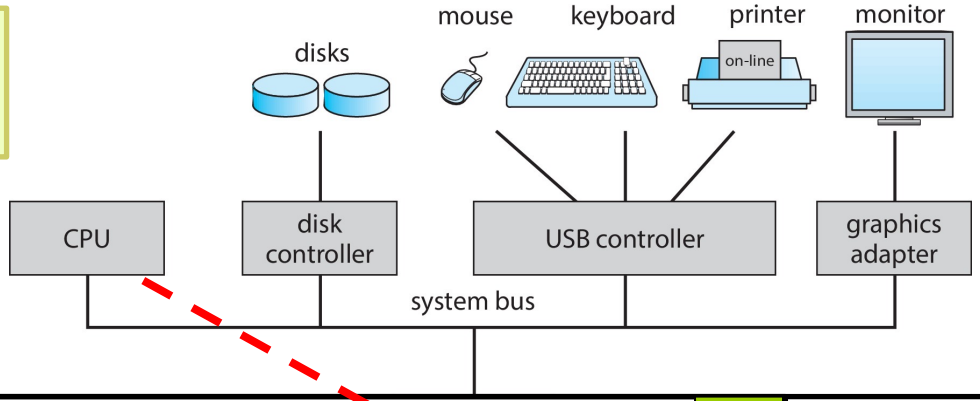
Application Program

…
Read file
SysCall()
…

-Read file Routine Implementation
-Write file
-Read net
Etc.

E.g. 1. Read Data From a File

ซีพียูรันคำสั่ง sys call เพื่ออ่านข้อมูลจากไฟล์



memory

OS kernel

Device Drivers
- Disk
- Mouse,
- KB,
- PRN
- GPU

System Call Interfaces

Application Program

…

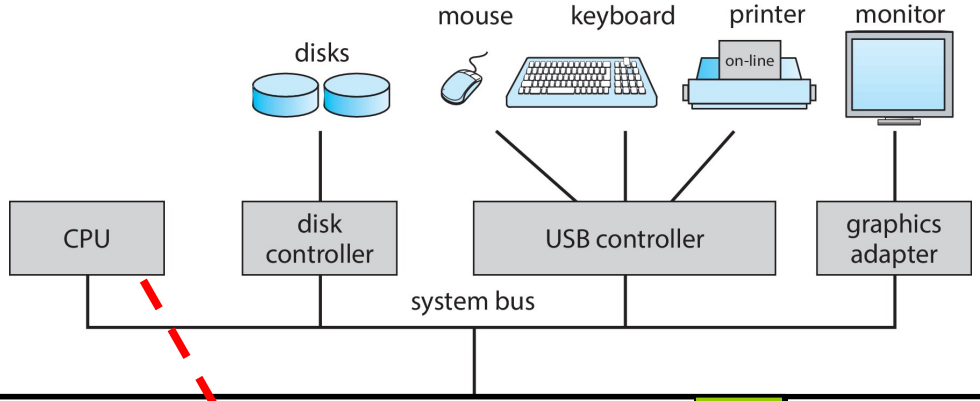
```
Read file
SysCall()
```

…

```
-Read file
Routine
Implementation
-Write file
-Read net
Etc.
```

E.g. Read Data From a File (1)

ซีพียูรันคำสั่ง ใน Implementation ของ sys call ใน OS Kernel



memory

OS kernel

Device Drivers
- Disk
- Mouse,
- KB,
- PRN
- GPU

-Read file Routine Implementation
-Write file
-Read net
Etc.

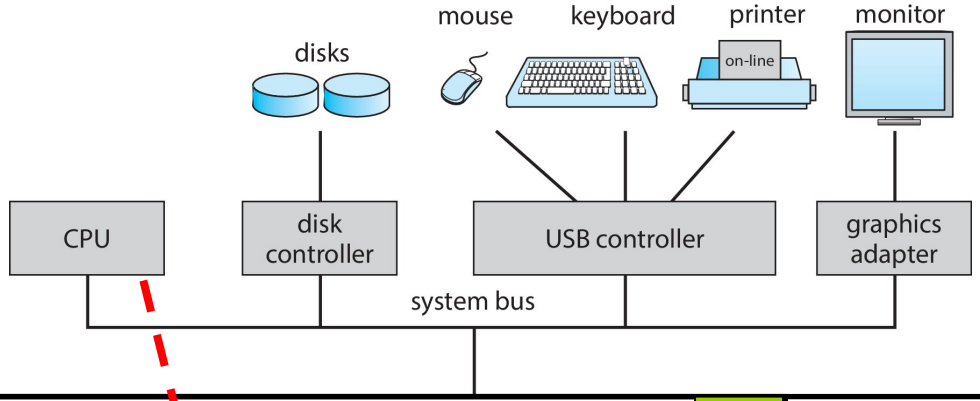
System Call Interfaces

Application Program

…
Read file
SysCall()
…

E.g. Read Data From a File (2)

ซีพียูรันคำสั่งเพื่อจัดการ Disk ใน Device driver



memory

OS kernel

Device Drivers
- Disk
- Mouse,
- KB,
- PRN
- GPU

-Read file Routine Implementation
-Write file
-Read net
Etc.

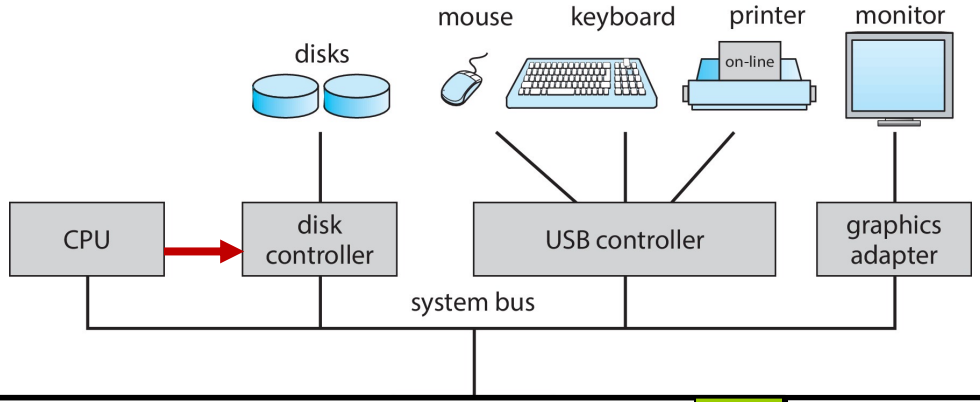
System Call Interfaces

Application Program

…
Read file
SysCall()
…

E.g. 1. Read Data From a File (3)

ซีพียูรันคำสั่งเพื่ออ่านข้อมูลจาก disk ส่งผลให้มีการสั่งงานไปที่ Disk controller



memory

OS kernel

Device Drivers
- Disk
- Mouse,
- KB,
- PRN
- GPU

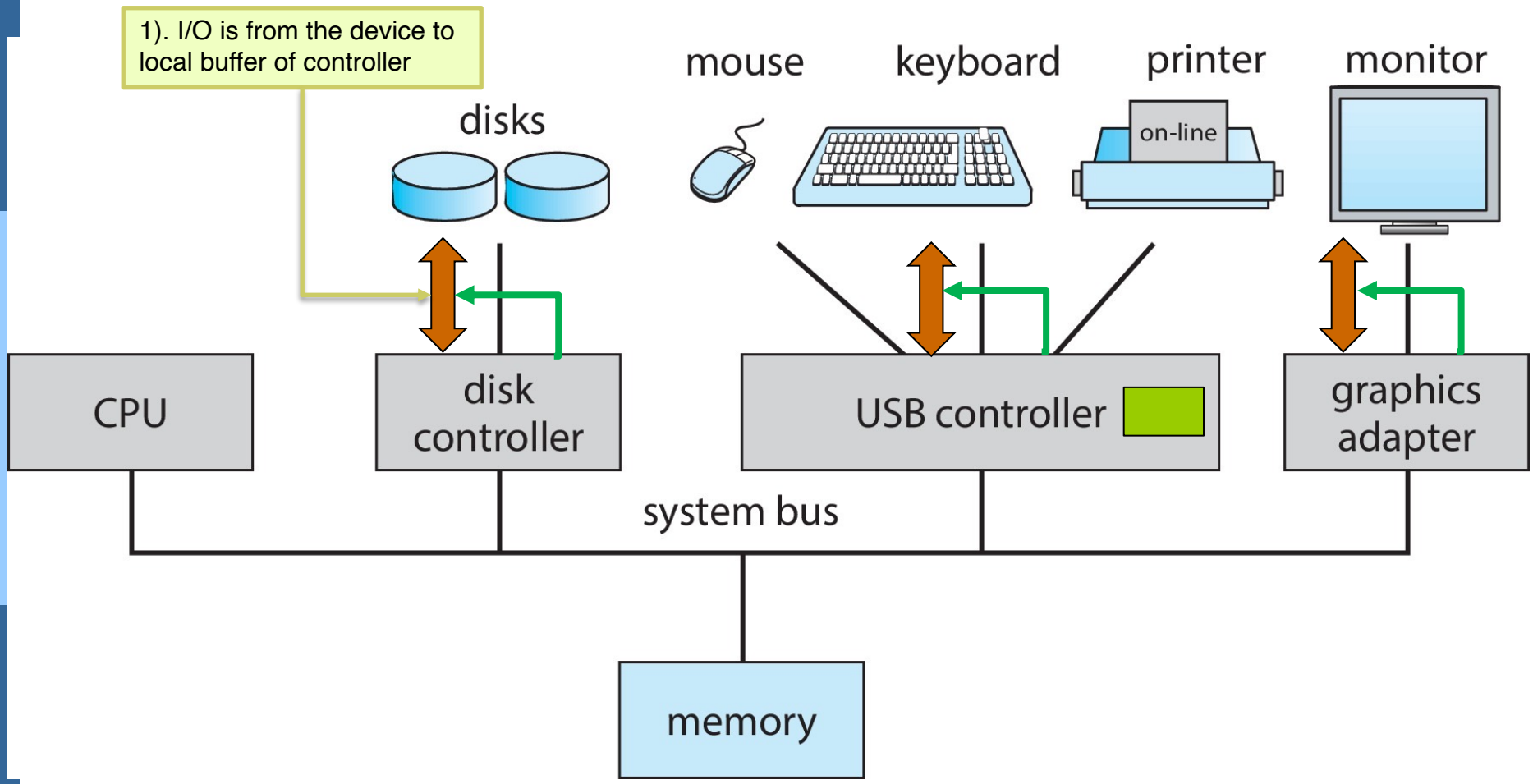
-Read file Routine Implementation
-Write file
-Read net
Etc.

System Call Interfaces

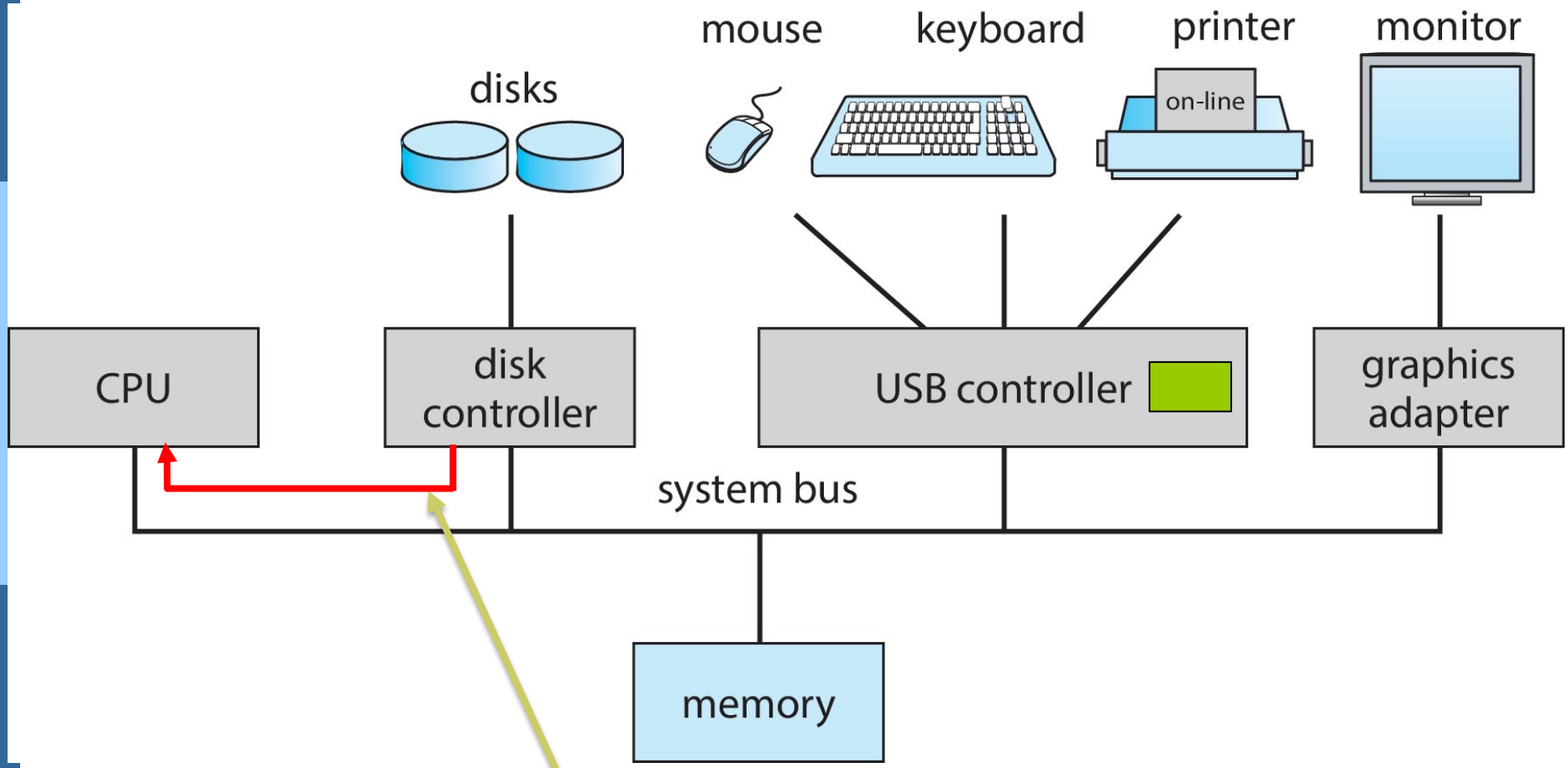
Application Program

…
Read file
SysCall()
…

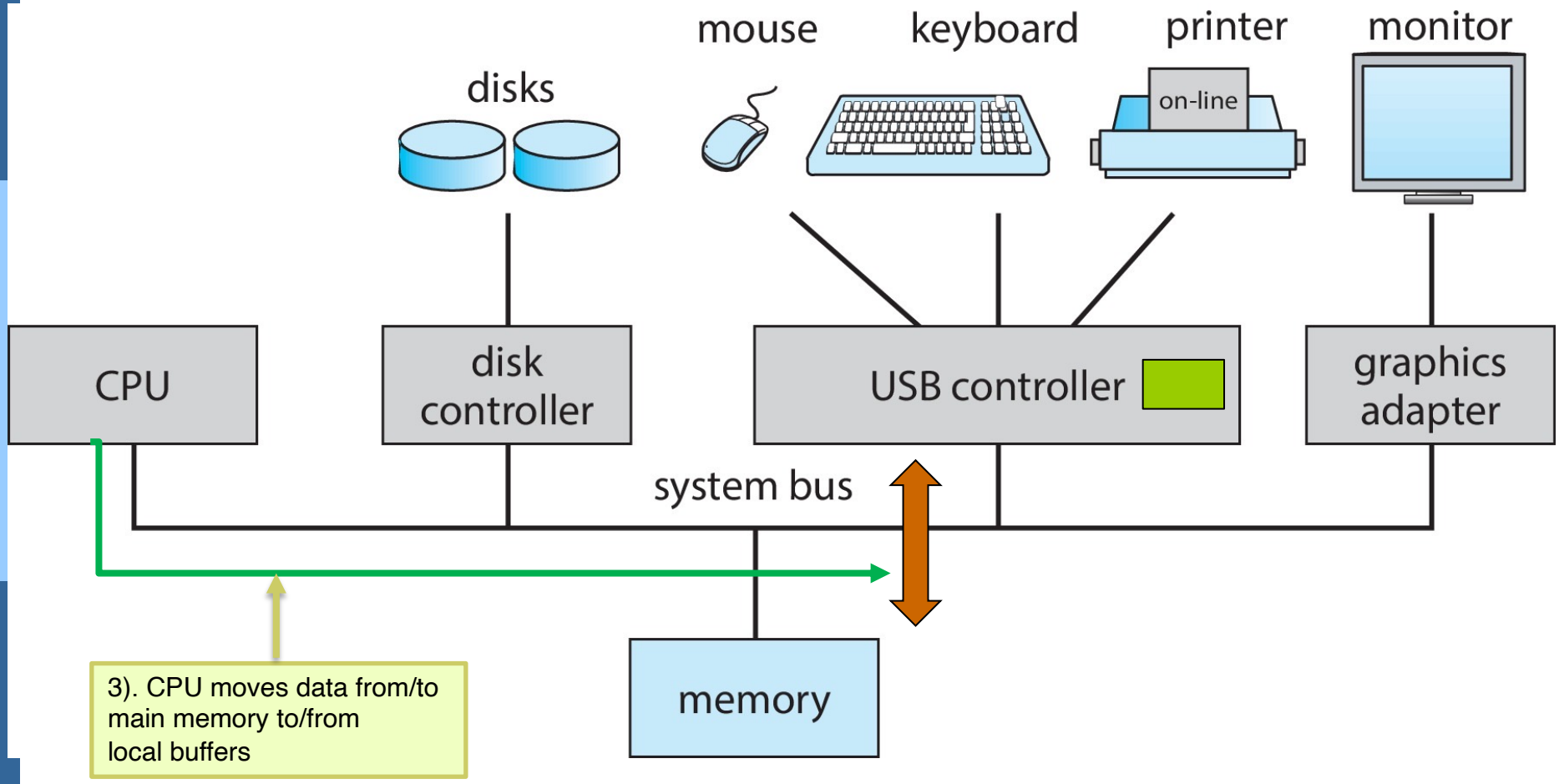
E.g. 1. Read Data From a File (4)



E.g. 1. Read Data From a File (5)



E.g. 1. Read Data From a File (6)



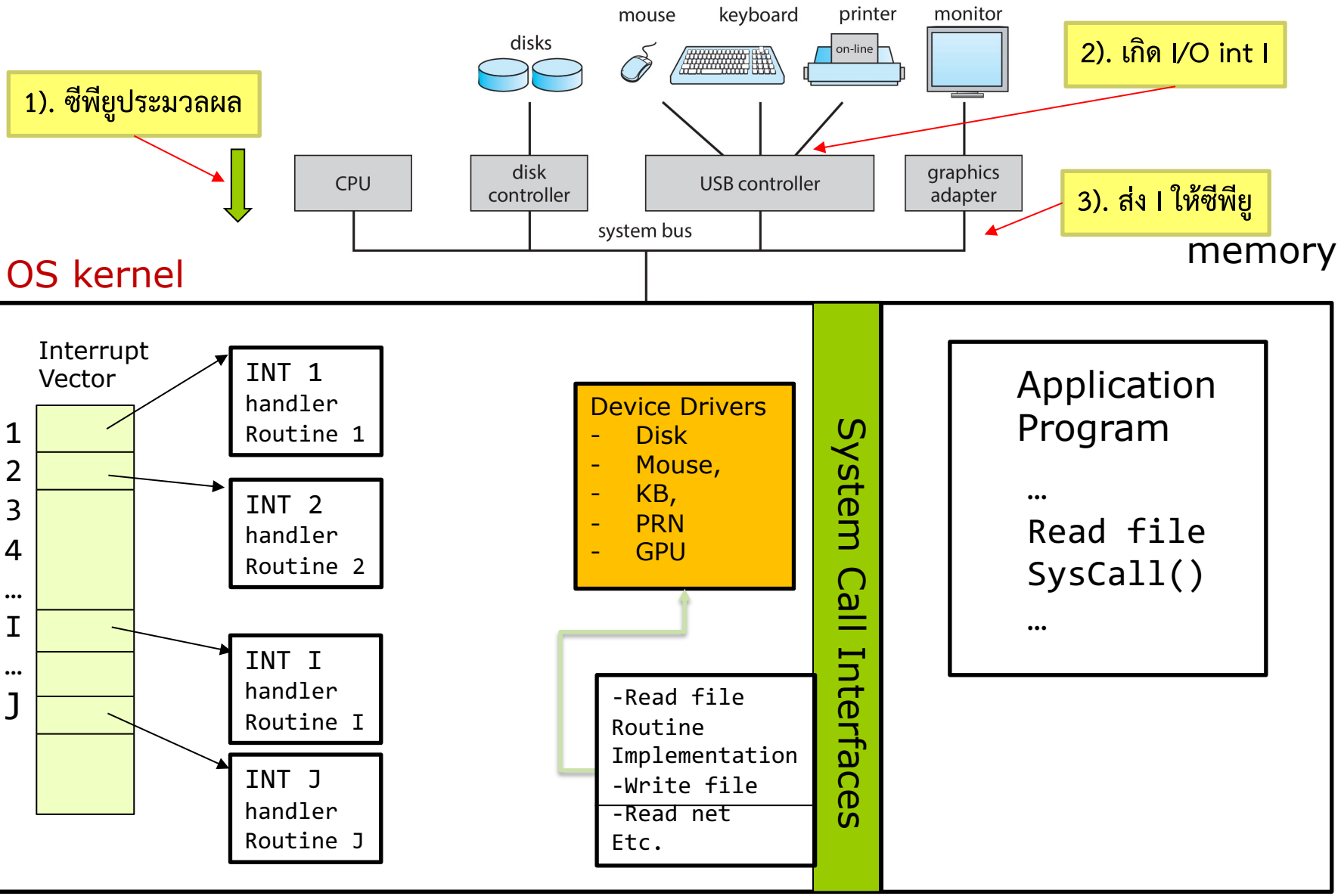
Common Functions of Interrupts (1)

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction
- An operating system is **interrupt driven**

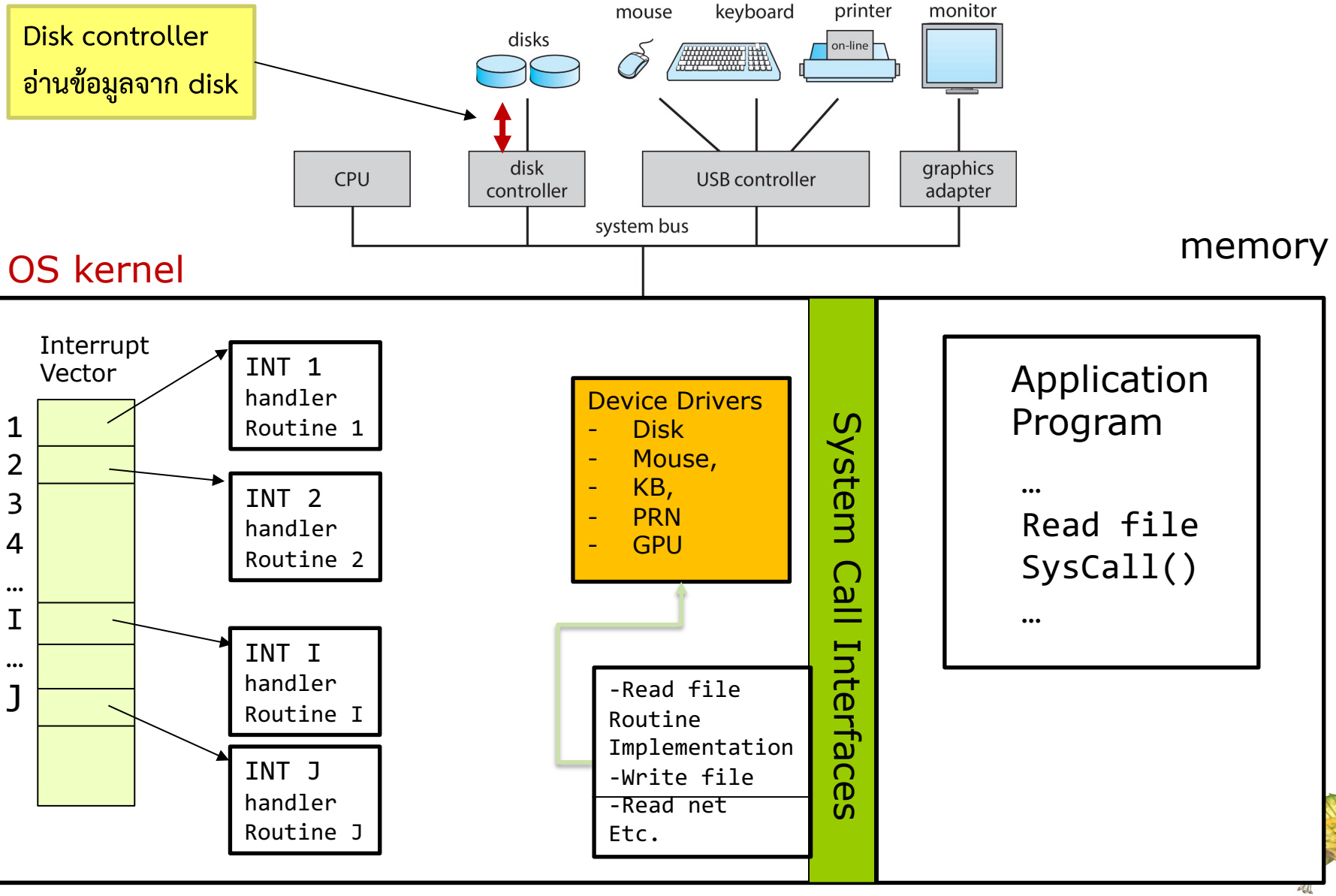
เหตุการณ์ขัดจังหวะในระบบคอมพิวเตอร์

- เมื่อมีเหตุการณ์เกิดขึ้น จะเกิดการส่งสัญญาณขัดจังหวะ (Interrupt) ไปยังซีพียู
- **Interrupt Number:** สัญญาณ Interrupt จะมีตัวเลข Integer ID กำกับเพื่อบอกชนิดของเหตุการณ์ที่เกิดขึ้น ตัวเลข ID นี้กำหนดโดยผู้ผลิต hardware และ OS
- **Interrupt Vector** คือ array ที่เก็บค่าของ memory address ของชุดคำสั่งสำหรับจัดการเหตุการณ์ที่ทำให้เกิดสัญญาณ Interrupt
- เมื่อซีพียู ถูกขัดจังหวะ มันจะนำ Interrupt Number ไปเทียบกับตารางใน Interrupt Vector และนำชุดคำสั่งของ **Interrupt Handler** ใน memory address ที่เก็บอยู่ในช่อง Interrupt Number ที่เกี่ยวข้องไปประมวลผล
- Interrupt Vector มีโครงสร้างที่ซับซ้อนขึ้นเมื่อต้องเก็บข้อมูล Handler สำหรับจัดการอุปกรณ์ I/O ที่มีจำนวนมาก ขอให้ศึกษาต่อใน Textbook

Interrupt Vector and Handler Routines

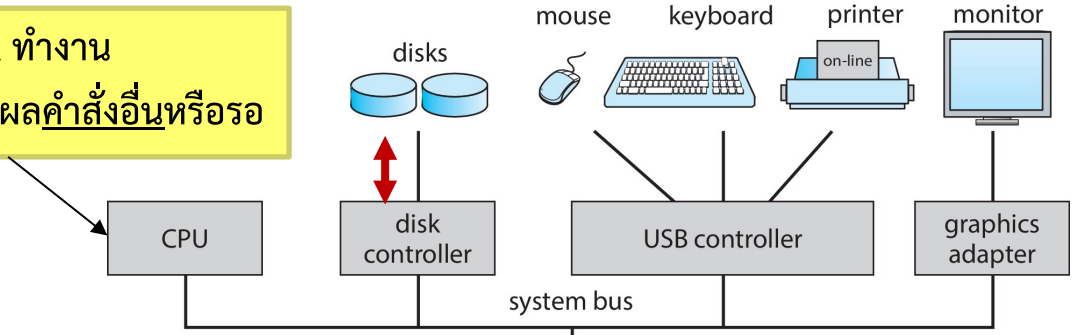


E.g. 1. Read Data From a File (4)



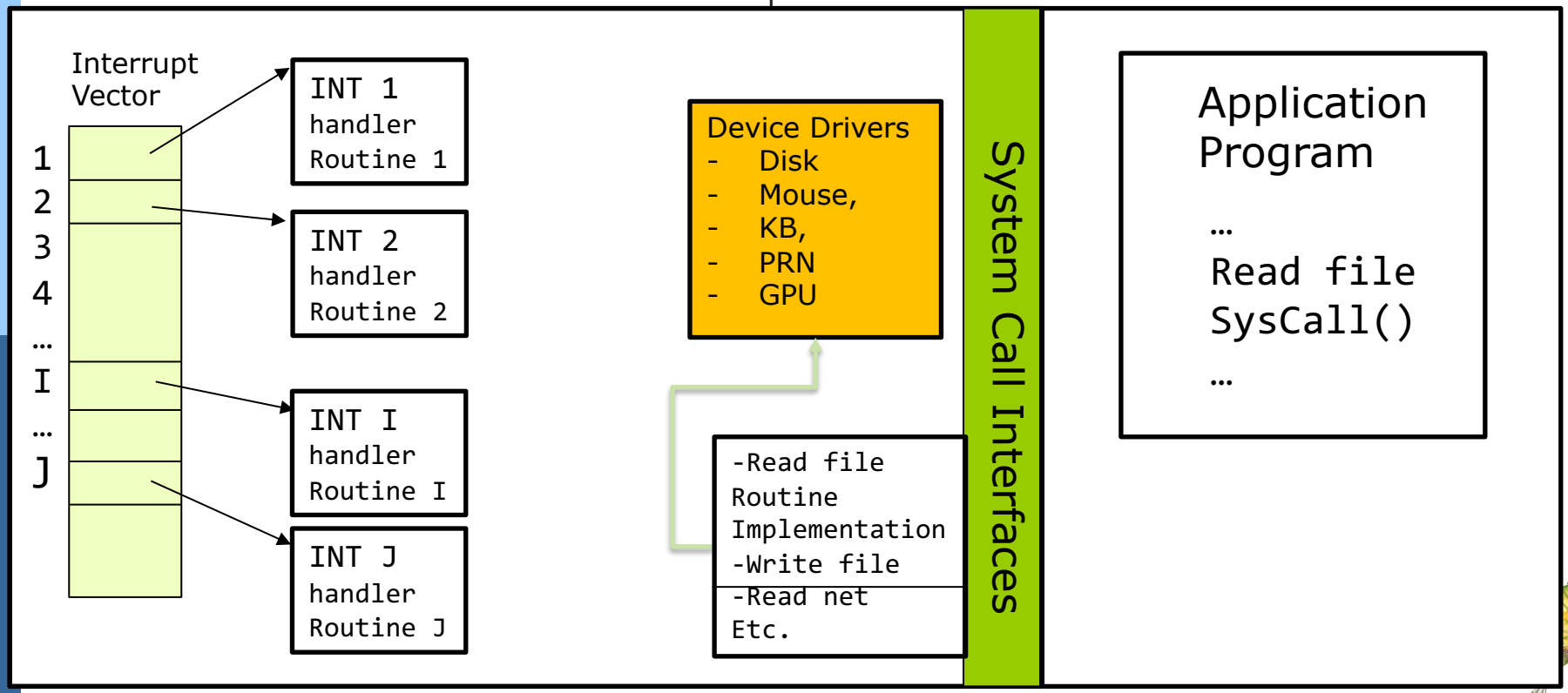
E.g. 1. Read Data From a File (5)

ในระหว่างที่ Disk ทำงาน
ซีพียูอาจประมวลผลคำสั่งอื่นหรือรอ



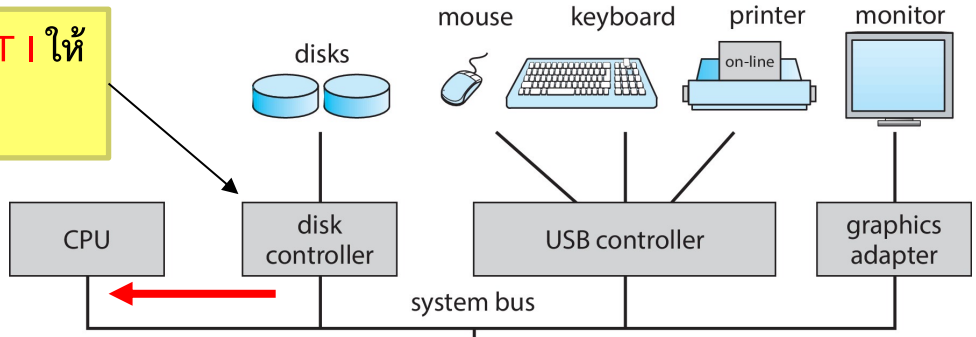
memory

OS kernel



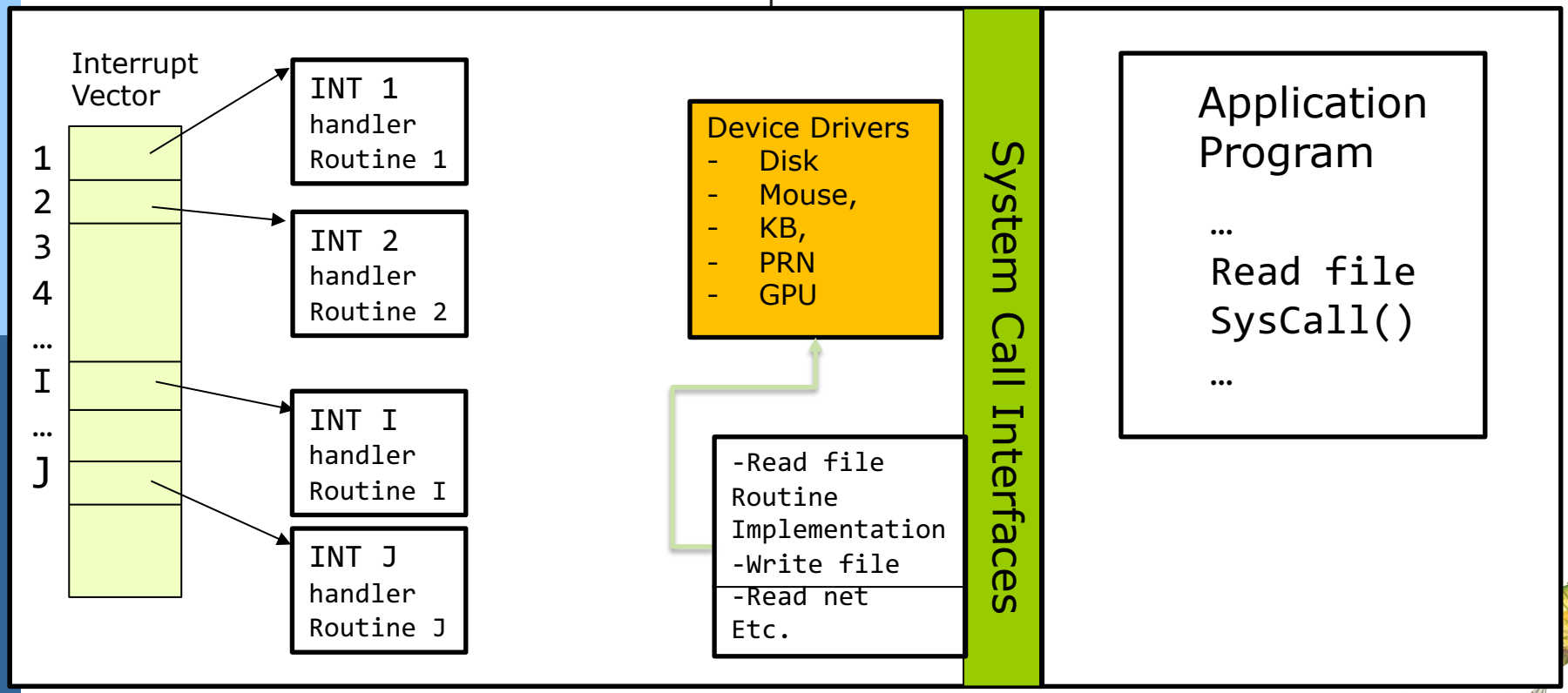
E.g. 1. Read Data From a File (6)

Disk controller ส่ง INT I ให้ซีพียูเมื่อทำงานเสร็จ



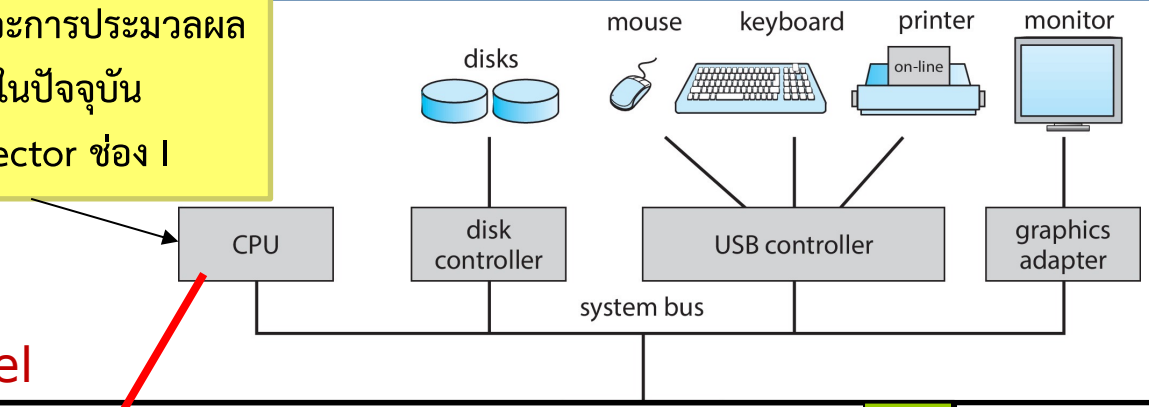
memory

OS kernel



E.g. 1. Read Data From a File (7)

ซีพียูขัดจังหวะการประมวลผลจากงานที่ทำในปัจจุบัน ไปเช็ค Int vector ช่อง I



memory

OS kernel

Interrupt Vector

1
2
3
4
…
I
…
J

INT 1 handler Routine 1

INT 2 handler Routine 2

INT I handler Routine I

INT J handler Routine J

Device Drivers
- Disk
- Mouse,
- KB,
- PRN
- GPU

-Read file Routine Implementation
-Write file
-Read net
Etc.

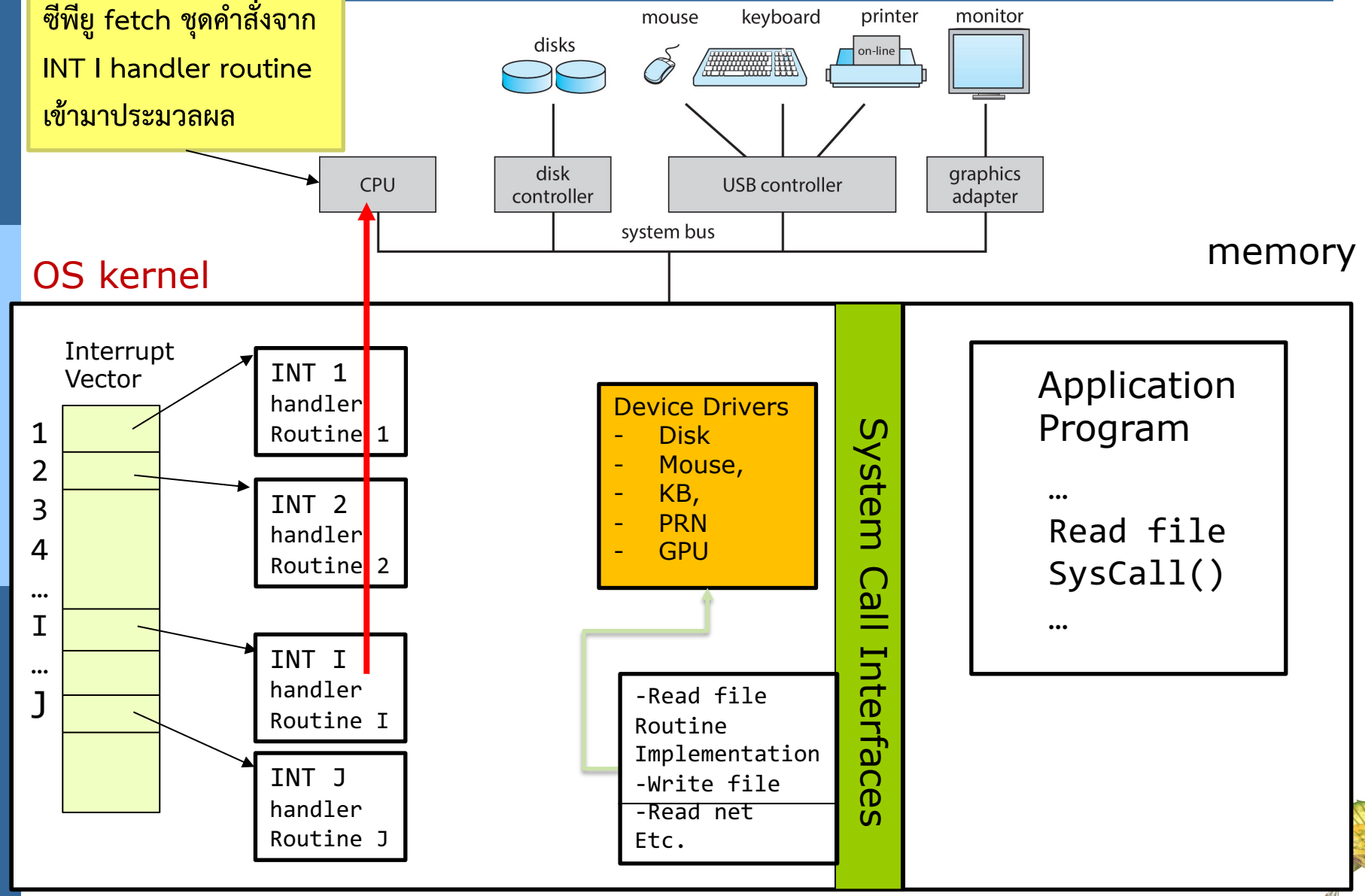
System Call Interfaces

Application Program

…
Read file
SysCall()
…

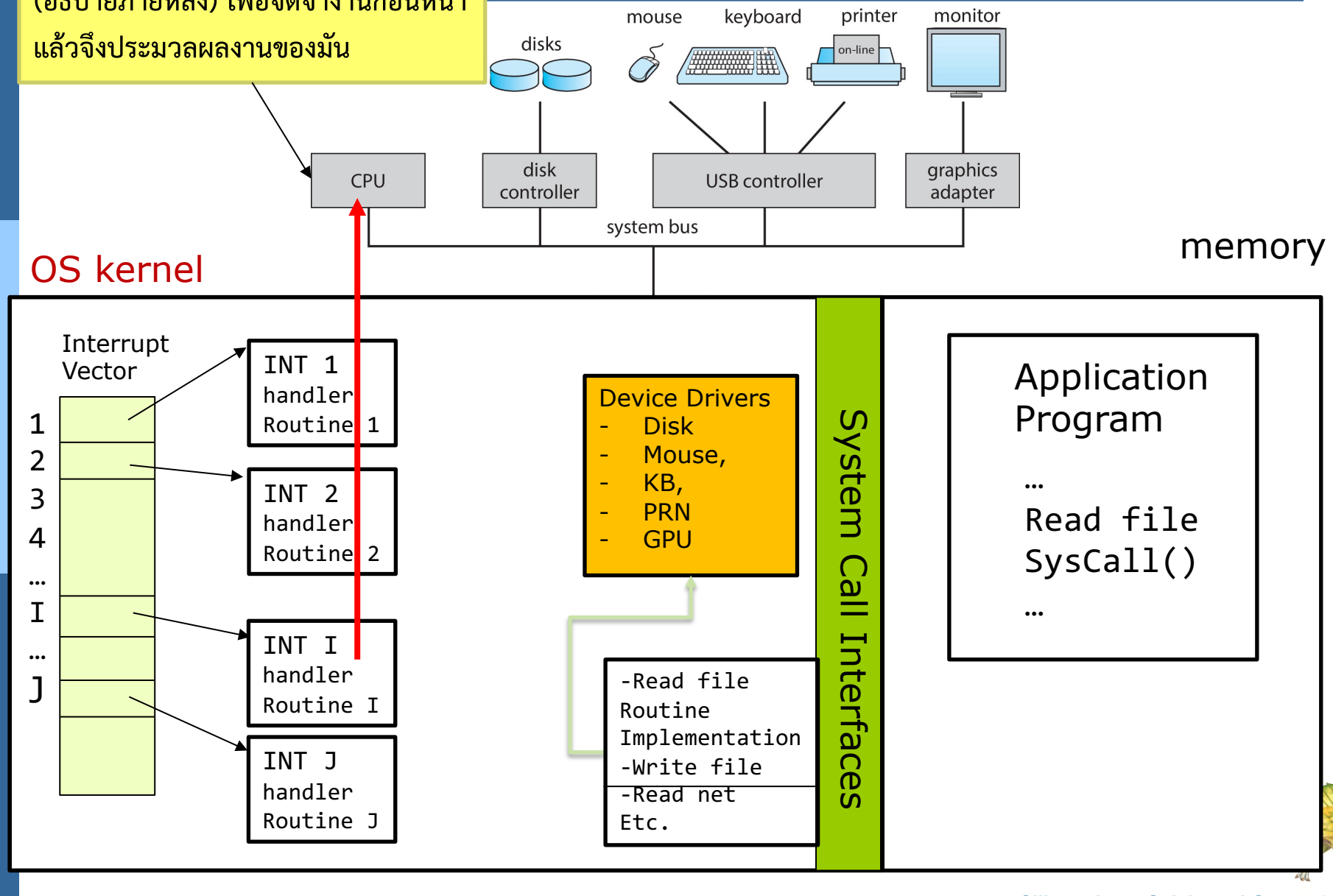
E.g. 1. Read Data From a File (8)

ซีพียู fetch ชุดคำสั่งจาก INT I handler routine เข้ามาประมวลผล



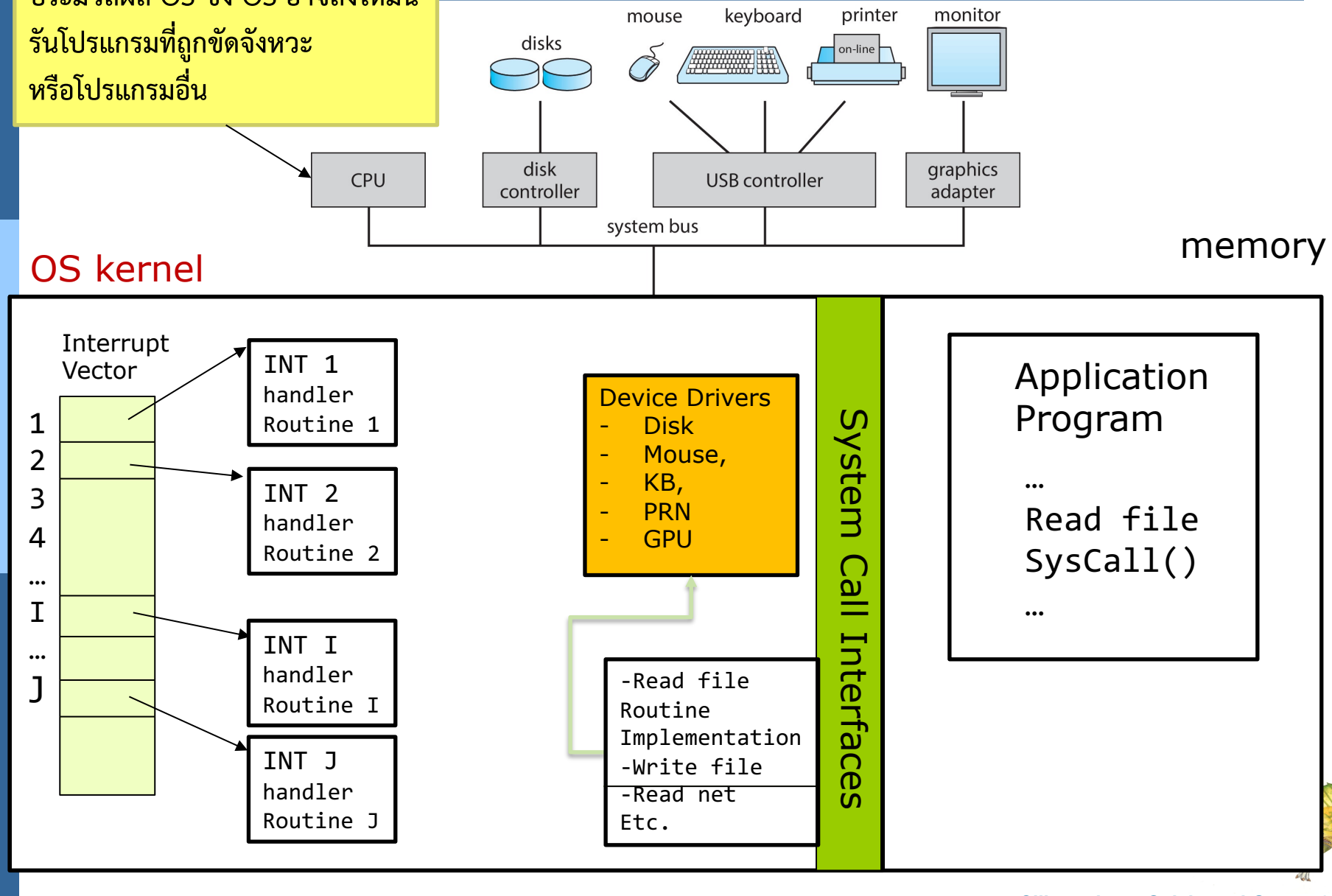
INT I handler routine
จะทำ context switching
(อธิบายภายหลัง) เพื่อจดจำงานก่อนหน้า
แล้วจึงประมวลผลงานของมัน

ead Data From a File (8)



เมื่อซีพียูเสร็จจากการประมวลผล INT handler ซีพียูจะกลับไปประมวลผล OS ซึ่ง OS อาจสั่งให้มันรันโปรแกรมที่ถูกขัดจังหวะหรือโปรแกรมอื่น

Read Data From a File (9)



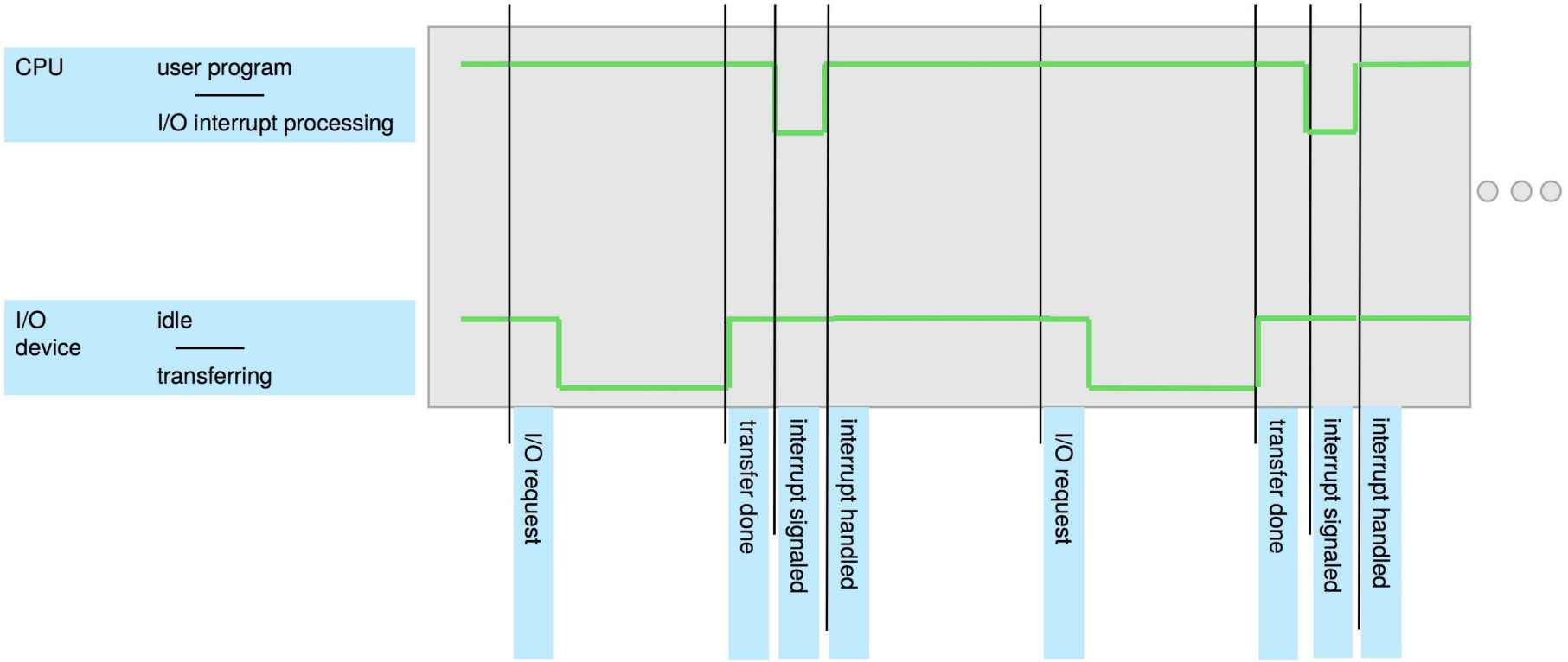
Common Functions of Interrupts (2)

- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request
- An operating system is **interrupt driven**

ชนิดของเหตุการณ์ขัดจังหวะ (ตามชนิดของการเกิด)

- ชนิดของ Interrupts เมื่อแบ่งตามชนิดของการเกิดจะมีสองประเภท
- ถ้าเหตุการณ์ขัดจังหวะซีพียูเกิดจาก **ภายนอกซีพียู** เช่น มีสัญญาณ ไอโอ เมื่อ Disk controller อ่านข้อมูลเสร็จ หรือเมื่อ Network Interface Controller ได้รับ network packet เป็นต้น เราเรียกว่า Interrupt
- ถ้าเหตุการขัดจังหวะซีพียู**เกิดเนื่องมาจากการประมวลผลของซีพียูเอง** เช่นเมื่อเกิดความผิดพลาดในการประมวลผล (หารตัวเลขด้วย 0, Arithmetic Overflow) เราเรียกการขัดจังหวะนั้นว่า Trap
- **ในบทต่อไป นศ จะเห็นว่าการทำงานของ OS นั้นถูกขับเคลื่อนโดย Interrupt อยู่เป็นประจำ (Interrupt-Driven)**

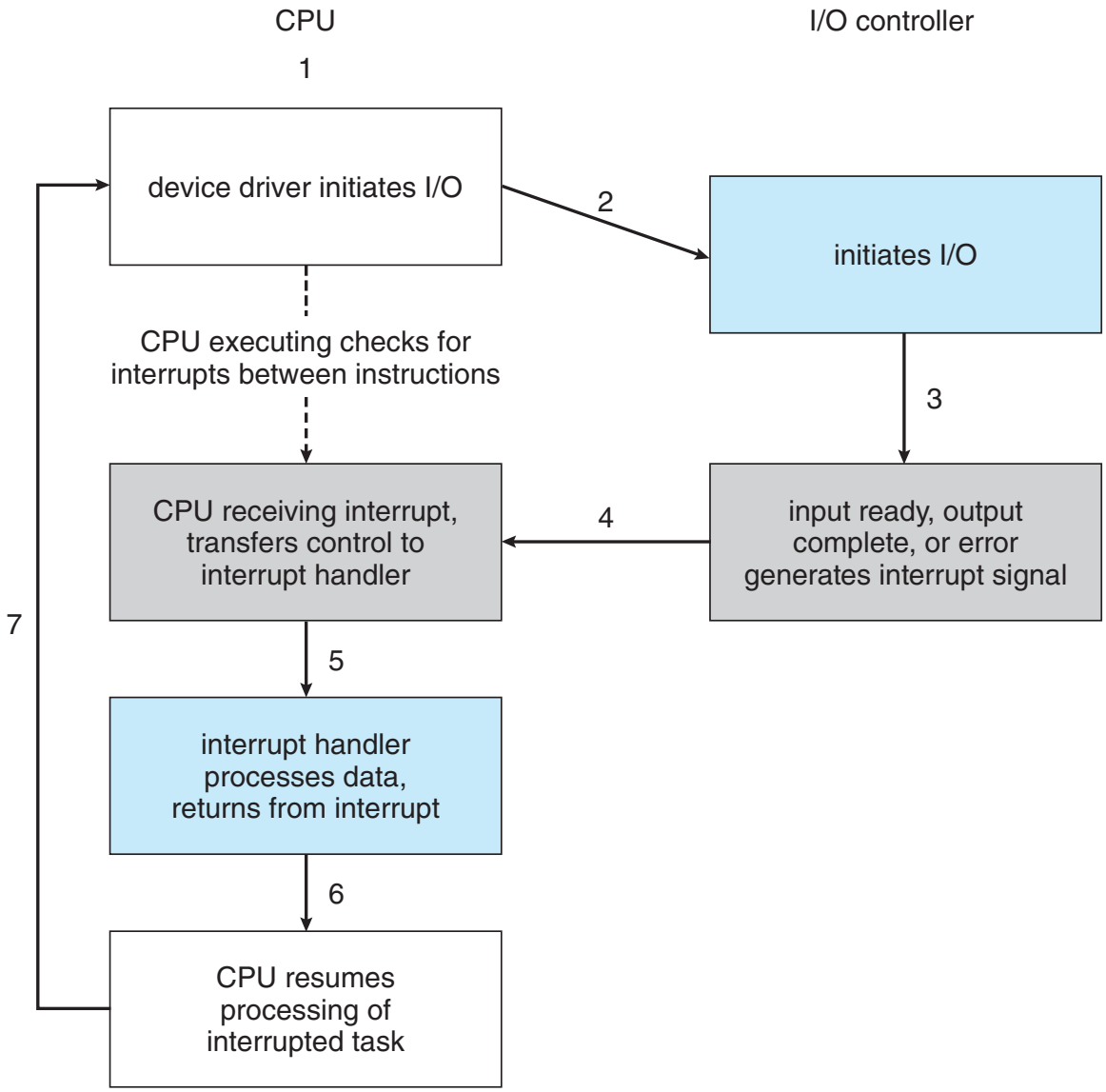
Interrupt Timeline



Interrupt Handling

- The operating system preserves the state of the CPU by storing the registers and the program counter
- Determines which type of interrupt has occurred:
- Separate segments of code determine what action should be taken for each type of interrupt

Interrupt-drive I/O Cycle

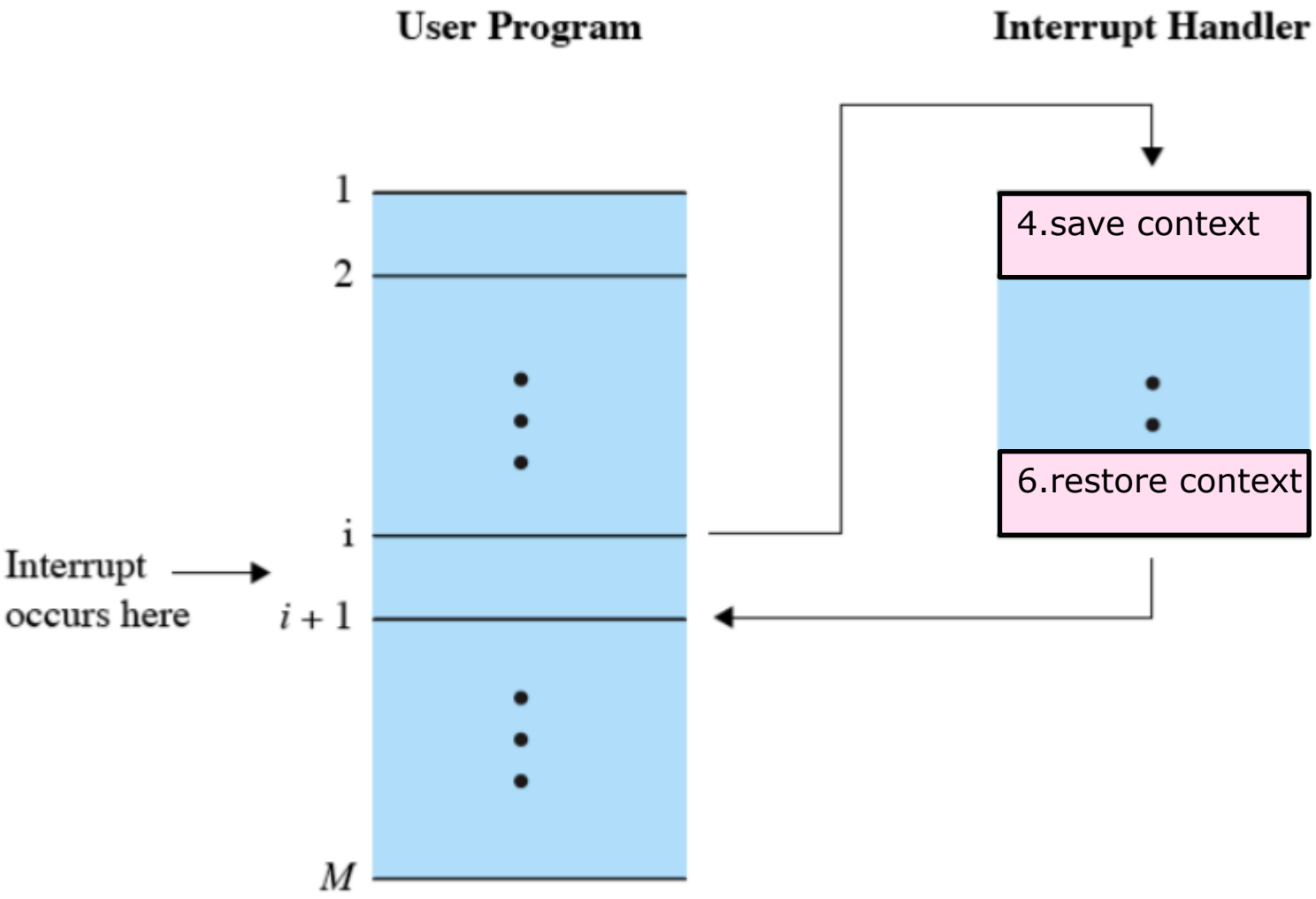


ซีพียูกลับไปทำงานเดิม ณ. คำสั่งที่ถูกขัดจังหวะได้อย่างไร

1. จะขอยกตัวอย่างจากสถานการณ์สมมุติ
2. ผู้จัดการร้านสะดวกซื้อกำลังเช็คสต๊อกสินค้าในร้านอยู่
3. ถูกขัดจังหวะเพราะมีลูกค้าเข้ามาซื้อของในร้าน
4. **ผู้จัดการ จดข้อมูลปัจจุบันเกี่ยวกับงานเช็คสต๊อกสินค้า ไว้ในสมุดจด**
5. **ผู้จัดการหันไปขายของให้ลูกค้า**
6. **เมื่อเสร็จจากการขายของ ผู้จัดการอ่านข้อมูลจากสมุดจดมาดูว่าก่อนถูกขัดจังหวะเช็คสต๊อกอะไรอยู่**
7. ผู้จัดการกลับไปทำงานเช็คสต๊อกต่อจากตอนที่ถูกขัดจังหวะ

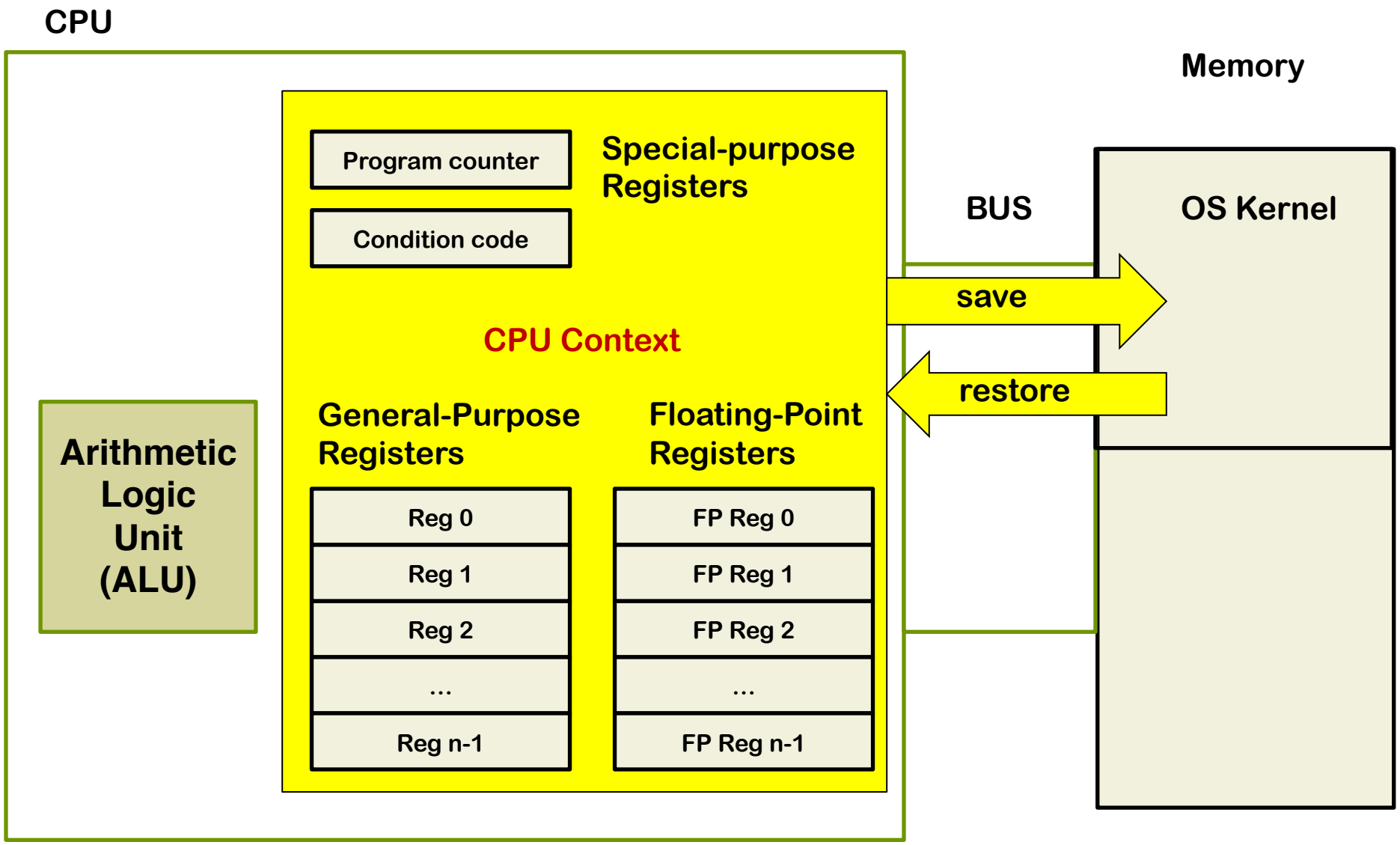
ซีพียูกลับไปทำงานเดิม ณ. คำสั่งที่ถูกขัดจังหวะได้อย่างไร

1. ในกรณีของซีพียู
2. ซีพียูกำลังประมวลผลโปรแกรมหนึ่งอยู่
3. ซีพียูถูกขัดจังหวะเพราะได้รับสัญญาณ Interrupt และรัน Interrupt Handler
4. **ใน Interrupt Handler: ซีพียู save ค่าของ Registers (CPU context) สำหรับการประมวลผลทั้งหมดในปัจจุบัน ไปเก็บในพื้นที่หน่วยความจำของ OS Kernel**
5. **ใน Interrupt Handler: ซีพียู ประมวลผลชุดคำสั่งของ Interrupt Handler Routine**
6. **ใน Interrupt Handler: เมื่อเสร็จ ซีพียูจะอ่าน CPU context จากหน่วยความจำของ OS kernel กลับเข้ามาใส่ใน registers ของซีพียู (รวมทั้ง Program Counter register)**
7. ซีพียู ประมวลผลคำสั่งของโปรแกรมที่ถูกขัดจังหวะต่อไป



Reference: William Stalling, Operating Systems: Internal and Design Principles

CPU Context Switching



CPU Context Switching

1. CPU context swithing = ขั้นตอน 4 + 6
2. ซีพียูต้องใช้เวลาในการประมวลผล CPU context switching
3. เมื่อ ซีพียูกลับมาประมวลผล ซีพียูอาจ (1) ประมวลผลคำสั่งถัดไปจากคำสั่งที่ถูกขัดจังหวะ หรือ (2) ประมวลผลคำสั่งเดิมที่ถูกขัดจังหวะ แล้วแต่เหตุการณ์ interrupt ที่เกิดขึ้น

ชนิดของการขัดจังหวะ (ตามความสำคัญของเหตุการณ์)

- การขัดจังหวะที่หลีกเลี่ยงไม่ได้ เรียกว่า non-maskable interrupts
- การขัดจังหวะที่จัดการได้ เรียกว่า maskable interrupts
- ในกรณีที่มี interrupt หลาย interrupt เกิดขึ้นพร้อมกัน CPU จะจัดการตาม Priority ของ interrupts เหล่านั้น

Interrupt vector table

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Figure 1.5 Intel processor event-vector table.

สรุป

- ทบทวนการประมวลผลของ CPU
- แสดงองค์ประกอบของระบบคอมพิวเตอร์
- แสดงการประมวลผลของซีพียูแบบมี Interrupt
- OS คือโปรแกรมที่มีรูปแบบการประมวลผลแบบ Interrupt Driven

Interrupt Controller

- Hardware ที่ทำหน้าที่จัดการสัณญาณอินเตอร์รับมีสองชนิด
- Local Advanced Programmable Interrupt Controller (LAPIC) เป็นอุปกรณ์จัดการสัญญาณอินเตอร์รับสำหรับแต่ละซีพียูคอร์ ส่ง timer interrupt ให้โลคอลซีพียูคอร์ และส่ง IPI ให้ LAPIC ของซีพียูคอร์อื่น
- I/O Advanced Programmable Interrupt Controller (IOAPIC) เป็นอุปกรณ์จัดการสัญญาณอินเตอร์รับจากไอโอสับซิสเต็ม (I/O subsystem) และส่งต่อให้ซีพียูคอร์ที่เป็นผู้รับ

IPI

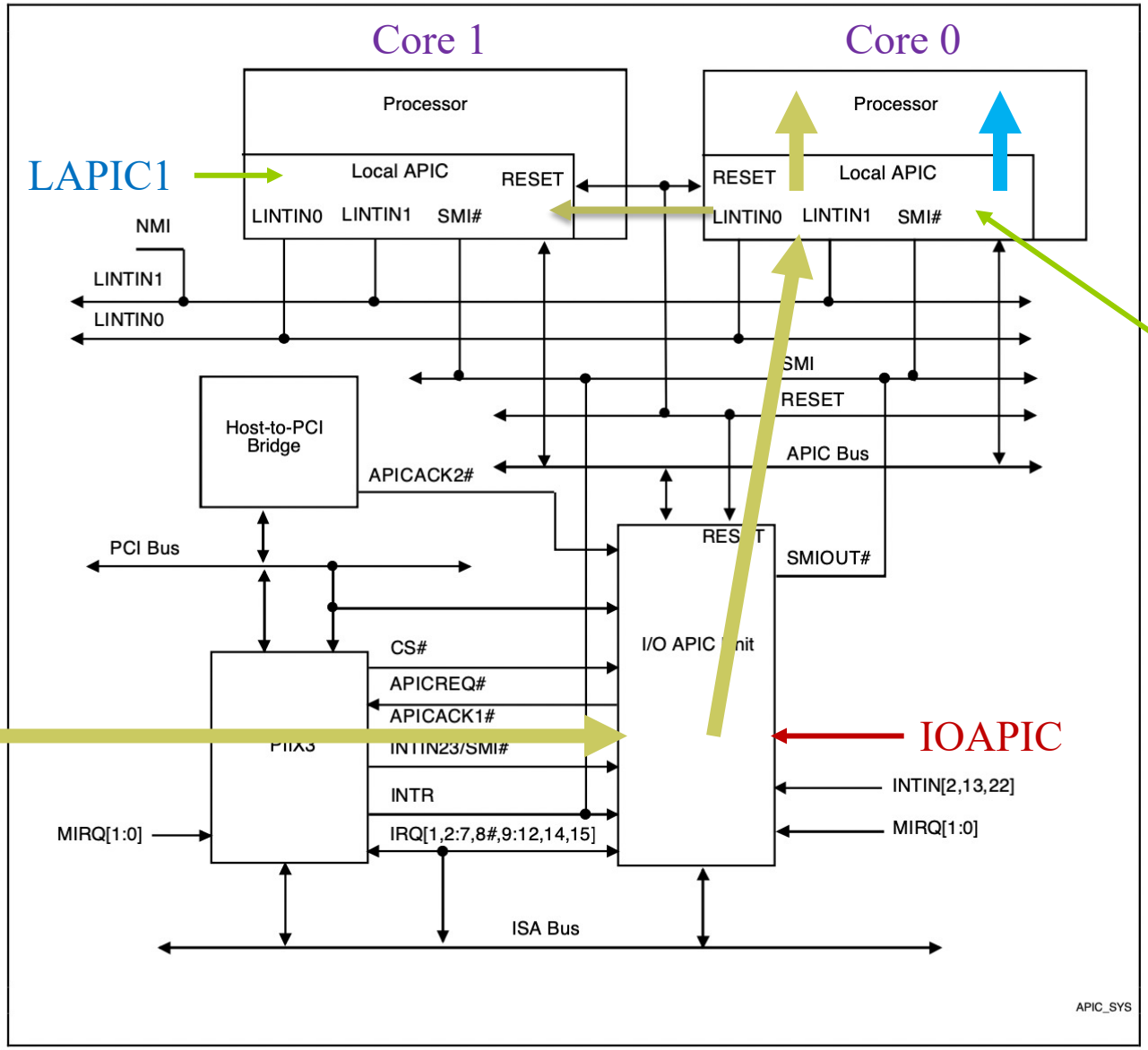


Figure 2. I/O And Local APIC Units

Maskable Interrupt Signals

- Maskable Interrupt (IRQ) คือสัญญาน Interrupt ที่ Programmer สามารถเขียนโปรแกรมให้ OS หรือ Process เลือกที่จะสนใจหรือไม่สนใจได้
  - เนื่องจากในระหว่างที่ OS หรือ Process รัน อาจมี Interrupt เกิดขึ้นมากมาย ดังนั้นจึงมีความจำเป็นต้องสามารถออกคำสั่งให้มีการจัดการ Interrupt เหล่านั้นในอันดับและเวลาที่เหมาะสม
- มี Hardware ควบคุมได้แก่ Programmable และ Advanced Programmable Interrupt Controller (PIC และ APIC)

Interrupt vector table

Article Talk

From Wikipedia, the free encyclopedia

This article is about the general concept. For its implementation found in x86 processors, see .

An **interrupt vector table** (**IVT**) is a data structure that associates a list of interrupt handlers with a list of interrupt requests in a table of interrupt vectors. Each entry of the interrupt vector table, called an interrupt vector, is the address of an interrupt handler (also known as ISR). While the concept is common across processor architectures, IVTs may be implemented in architecture-specific fashions. For example, a dispatch table is one method of implementing an interrupt vector table.

Interrupts are assigned a number between 0 to 255. The interrupt vectors for each interrupt number are stored in the lower 1024 bytes of main memory. For example, interrupt 0 is stored from 0000:0000 to 0000:0003, interrupt 1 from 0000:0004 to 0000:0007, and so on.

Interrupt List

Interrupt Number	IVT Address	Interrupt Name
0	00-03	CPU divide by zero
1	04-07	Debug single step
2	08-0B	Non Maskable Interrupt (NMI input on processor)
3	0C-0F	Debug breakpoints
4	10-13	Arithmetic overflow
5	14-17	BIOS provided Print Screen routine
6	18-1B	Reserved
7	1C-1F	Reserved
8	20-23	IRQ0, Time of day hardware services
9	24-27	IRQ1, Keyboard Interface
A	28-2B	IRQ2, ISA Bus cascade services for second 8259
B	2C-2F	IRQ3, Com 2 hardware
C	30-33	IRQ4, Com1 hardware
D	34-37	IRQ5, LPT2, Parallel port hardware (Hard Disk on XT)
E	38-3B	IRQ6, Floppy Disk adaptor

F	3C-3F	IRQ7, LPT1, Parallel port hardware
10	40-43	Video services, see note 1
11	44-47	Equipment check
12	48-4B	Memory size determination
13	4C-4F	Floppy I/O routines
14	50-53	Serial port I/O routines
15	54-57	PC used for Cassette tape services
16	58-5B	Keyboard I/O routines
17	5C-5F	Printer I/O routines
18	60-63	Points to basic interpreter in a "real" IBM PC
19	64-67	Bootstrap loader
1A	68-6B	Time of day services
1B	6C-6F	Services Ctrl-Break service
1C	70-73	Timer tick (provides 18.2 ticks per second)
1D	74-77	Video parameters
1E	78-7B	Disk parameters
1F	7C-7F	Video graphics
20	80-83	Program termination (obsolete)
21	84-87	All DOS services available through this Interrupt
22	88-8B	Terminate address

23	8C-8F	Ctrl-Break exit address
24	90-93	Critical error handler
25	94-97	Read logical sectors
26	98-9B	Write logical sectors
27	9C-9F	Terminate and stay resident routines (obsolete)
28 to 3F	A0-A3 to FC-FF	Reserved for DOS
40 to 4F	100-103 to 13C-13F	Reserved for BIOS
50	140-143	Reserved for BIOS
51	144-147	Mouse functions
52 to 59	148-14B to 164-167	Reserved for BIOS
5A	168-16B	Reserved for BIOS

Traps v.s. Interrupts

- ในมิติของความสัมพันธ์กับยูเซอร์โปรแกรม เราแบ่งอินเตอร์รับเป็น 2 กลุ่ม
- Trap คือการที่ CPU ต้องเปลี่ยนการประมวลผลจากชุดคำสั่งของ user โปรแกรมที่กำลังประมวลผลในปัจจุบันไปประมวลผลชุดคำสั่งของ OS เพื่อตอบสนองกับ เหตุการณ์พิเศษที่เกิดขึ้นอันเป็นผลข้างเคียงของการประมวลผลคำสั่งของ user โปรแกรมนั้น
  - มักจะเกิดขึ้นเมื่อมี exception หรือ error ในการรันคำสั่งของโปรแกรม
  - User program สามารถเรียก system call เพื่อทำให้เกิด trap ขึ้นได้
  - E.g. Arith overflow, page faults, violation of privilege level, violation of mem prot., illegal opcode
- Interrupt เกิดจากเหตุการณ์ที่เกิดขึ้นบนระบบคอมพิวเตอร์ภายนอกการประมวลผล User โปแกรม
  - E.g. I/O interrupt and timer interrupts

Trap v.s. Interrupts

- ใครเป็นผู้ทำให้เกิดสัญญาน ขัดจังหวะการประมวลผลของ CPU?
- จากมุมมองของการปฏิบัติงาน ณ. เวลาปัจจุบันของ CPU ซึ่งจะต้องรัน Process ใด Process หนึ่ง (รวมทั้ง OS เพราะ OS ก็ถือเป็น Process เช่นกัน) ผู้ที่ทำให้เกิดสัญญานขัดจังหวะ CPU นั้นก็มีได้ 2 แบบได้แก่
  - เกิดจากคำสั่งของ Process นั้นเอง: เราเรียกสัญญาน ขัดจังหวะที่เกิดจาก การประมวลผลของ Process ที่รันอยู่ว่า Trap
  - เกิดจากสิ่งแวดล้อมอื่นๆที่อยู่ภายนอก: เราเรียกสัญญานแบบนี้ว่า Interrupts

Examples of Trap (1)

- เป็นเหตุการแบบ Synchronous คือ Trap เกิดขึ้นเพราะการรันคำสั่งใดคำสั่งหนึ่งของ Process และคำสั่งนั้นต้องรอรับผลจากการเกิด Trap
- System Call: โปรแกรมรันคำสั่งพิเศษเช่น SYSCALL/SYSRET หรือ INT (x86) เพื่อเปลี่ยน Privilege เป็น System mode เพื่อใช้บริการ services ต่างๆของ OS เพื่อบริหารจัดการทรัพยากรของคอมพิวเตอร์
- Arithmetic Overflow: การเพิ่มหรือลดค่าของตัวเลขมากกว่าที่ตัวแปรชนิดที่ใช้อยู่จะแทนค่าได้ หรือค่าที่มากกว่าที่การประมวลผลตัวเลขจะทำได้
- Arithmetic Error: การหารเลขด้วย 0

Examples of Trap (2)

- Page Fault: Memory page ที่ต้องการใช้งานไม่อยู่ใน Physical Memory
- Violation of CPU Privilege Levels: การรัน System ISA ในขณะที่ CPU ประมวลผลใน User Mode
- Violation of Memory Protection: การพยายามเปลี่ยนแปลงค่าใน Memory ส่วนที่ไม่ได้รับอนุญาตให้เปลี่ยนแปลงค่าเช่น Code Segment
- Illegal Opcode: การส่งรหัสคำสั่ง binary ที่ไม่ใช่ชุดคำสั่งที่ถูกต้องให้ CPU

Examples of Interrupts

- มักจะเป็นเหตุการณ์ที่เกิดแบบ Asynchronous คือสามารถเกิดเมื่อไรก็ได้ในระหว่างที่ Process กำลังประมวลผลคำสั่งใดๆก็ได้
- System Timer Interrupt: เมื่อ Timer นับถอยหลังถึง 0
- I/O devices: เมื่อมีการกด Keyboard, เคลื่อน Mouse,
- Disk Interrupts: ส่งสัญญานเมื่อการอ่านข้อมูลจาก Disk หรือเขียนข้อมูลลง Disk เสร็จสิ้น
- Network Interrupts: ส่งสัญญานเมื่อการรับข้อมูลจาก Network หรือส่งข้อมูลสู่ Network เสร็จสิ้น
- NMI Interrupts ที่ยกตัวอย่างไปก่อนหน้า