



การจัดองค์การคอมพิวเตอร์

พ4.2 ส่วนประกอบพื้นฐานของ ภาษาเครื่อง

31110321 Computer Organization

สำหรับนักศึกษาชั้นปีที่ 3 สาขาวิชาวิศวกรรมคอมพิวเตอร์

ทรงฤทธิ์ กิตติศรีวรพันธุ์

songrit@npu.ac.th

สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยนครพนม

Lecture plan

- 4.1 ภาษาเครื่อง
- **4.2 ส่วนประกอบพื้นฐานของภาษาเครื่อง**
- 4.3 ระบบแอสเซมบลีคอมพิวเตอร์และภาษาเครื่อง
- 4.4 ภาษาเครื่องแอสเซมบลี
- 4.5 อินพุต / เอาต์พุต
- 4.6 การเขียนโปรแกรมสำหรับเครื่องแอสเซมบลี
- 4.7 ภาพรวมโปรเจกต์ #4

ส่วนที่คำนึงถึง ภาษาเครื่อง

- ทราบคุณสมบัติ ฮาร์ดแวร์ / ซอฟต์แวร์
 - หน้าที่ของคำสั่งนั้น
 - ติดต่อกับส่วนใดบ้าง
 - ควบคุมโปรแกรมอย่างไร
- โดยปกติเป็นคำสั่งมีผลโดยต่อกับชิปต่างๆ
 - การคำนวณ , บวก , ลบ
 - การทำเงื่อนไข
 - การควบคุมลำดับการทำงาน : goto คำสั่ง เมื่อ $A == B$
- เทียบ ค่าใช้จ่าย กับ ประสิทธิภาพที่ได้
 - ปริมาณชิปที่ต้องการใช้
 - เวลาในการรันโปรแกรม

ส่วนที่คำนึงถึง ภาษาเครื่อง (ต่อ)

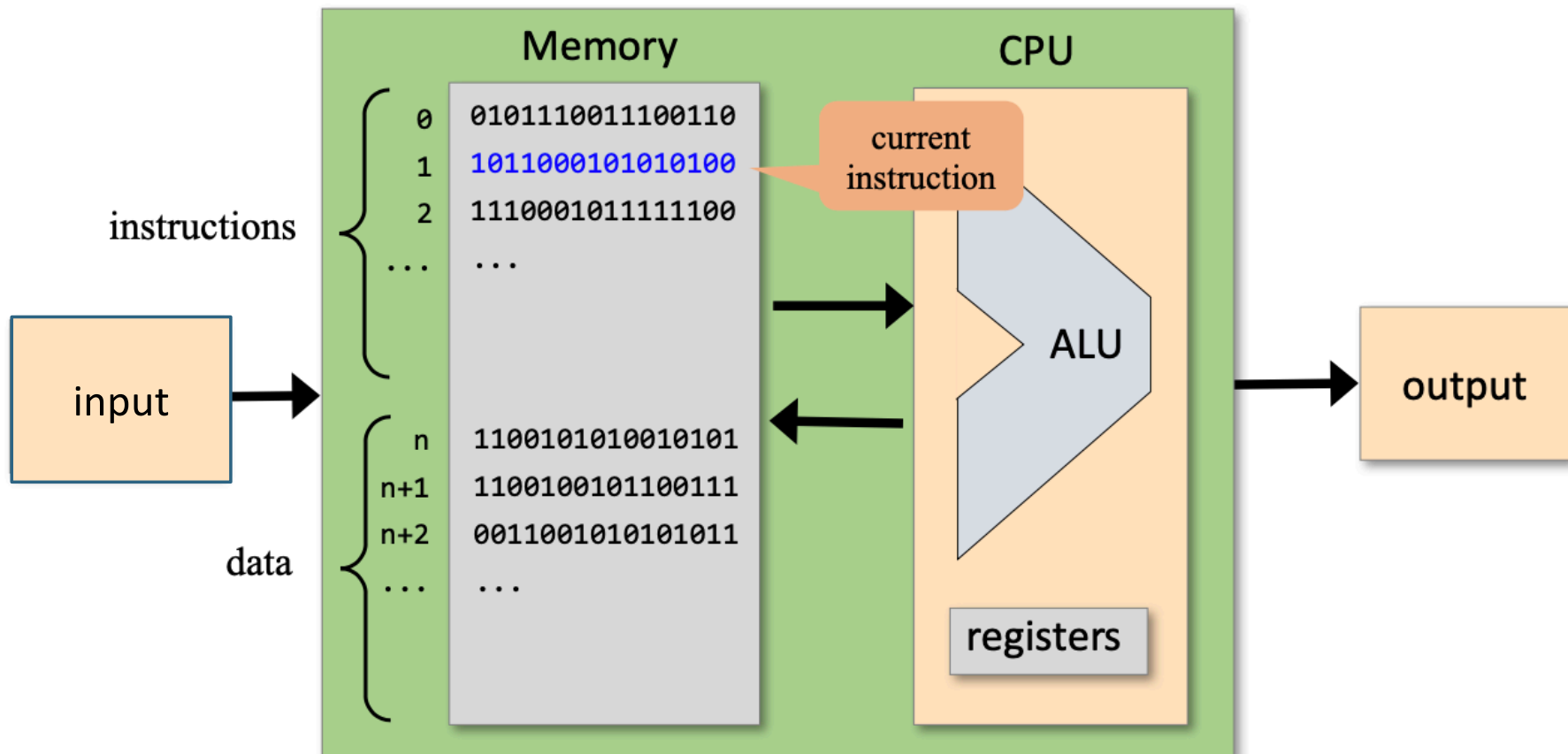
- คำสั่งภาษาเครื่องมาก
 - มีคำสั่งให้เลือกใช้จำนวนมาก (การหาร , การ copy เป็นช)
 - ชนิดข้อมูลหลากหลาย (float, double, strings)

Basic elements

- ภาษาเครื่อง
 - คำสั่งไบนารี ควบคุมชิปเซต
- ลำดับการรับคำสั่ง
 - คำสั่งด้านลอจิก
 - ADD, subtract, ...
 - AND, OR , NOT, ...
 - Flow control : goto , if , loop
 - คำสั่งพิเศษ
 - คำสั่งที่ใช้หลายขั้นตอน (การหาร การคูณ)
 - การคำนวณเลขจำนวนจริง
- ชุดคำสั่ง และ ข้อมูล ถูกเก็บในหน่วยความจำ

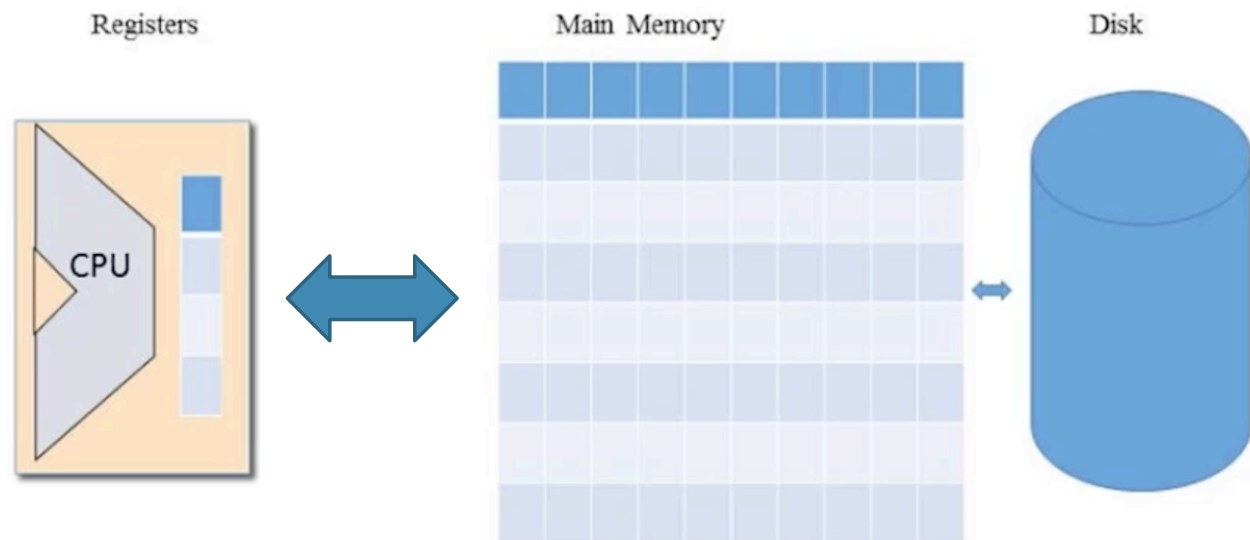
Addressing

- คำสั่ง/ข้อมูล เก็บในหน่วยความจำ
- เรียกตำแหน่งคำสั่งว่า แอดเดรส



Memory Hierachy

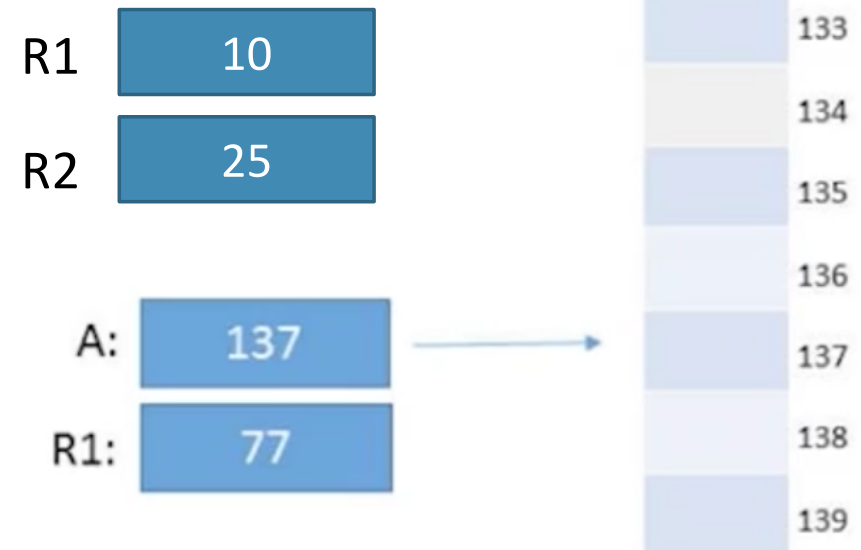
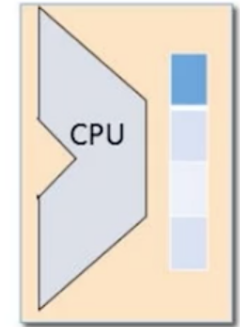
- หน่วยความจำหลักทำงานช้า
 - ต้องการการระบุ**แอดเดรส**หน่วยความจำ หลายแอดเดรส
 - ใช้การประมวลผลมาก
- ปรับประสิทธิภาพ : memory hierarchy



เนื้อที่เก็บข้อมูลขนาดเล็ก ทำให้เข้าถึงข้อมูลได้เร็ว

หน่วยความจำ แคช

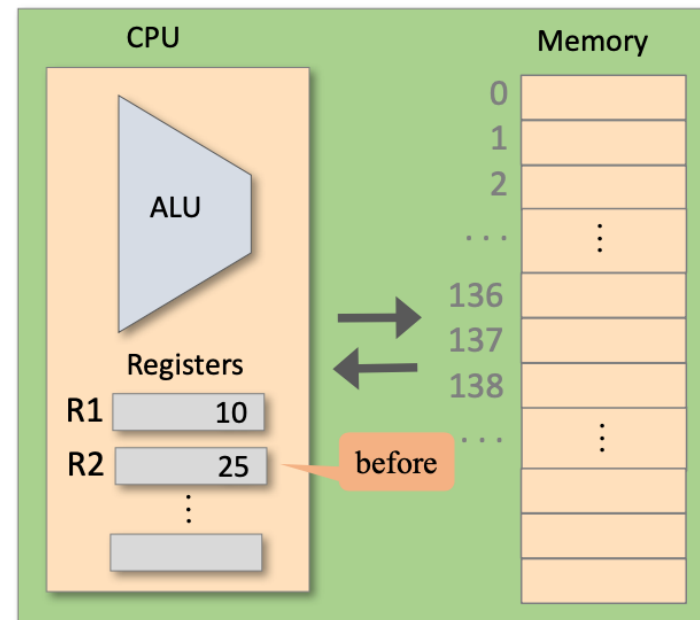
- ภายในซีพียูมีหน่วยความจำขนาดเล็ก เรียกว่า **รีจิสเตอร์** (registers)
- มีรีจิสเตอร์ไม่กี่ตัวภายในซีพียู
- Data Registers (เร็ว)
 - Add R1, R2
- Address Registers (ช้า)
 - Store R1, @A



Registers

- Registers เป็นหน่วยความจำขนาดเล็กใน CPU
- รีจิสเตอร์มีแอดเดรสเดียว
- การเขียน/อ่าน ทำได้เร็วกว่า การติดต่อหน่วยความจำ

- Data registers:
 - add R1, R2



Addressing Modes

- Register

- Add R1, R2 //R2 \leftarrow R2 + R1

- Direct

- Add R1, M[200] // Mem[200] \leftarrow Mem[200]+R1

- Indirect

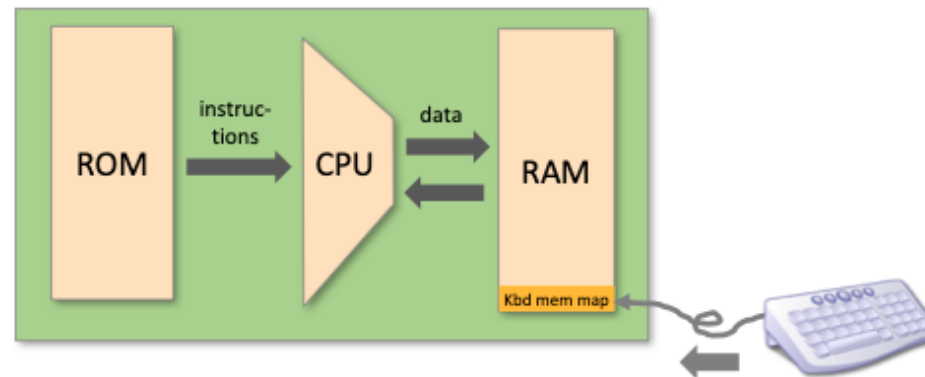
- Add R1, @A // Mem[A] \leftarrow Mem[A] + R1

- Immediate

- Add 73, R1 // R1 \leftarrow R1 + 73

Input / Output

- อุปกรณ์ต่อคอมพิวเตอร์ แบ่งเป็น 2 กลุ่ม Input และ Output
 - **Input** : คีย์บอร์ด, เมาท์ , ทัชสกรีน
 - **Output** : จอภาพ, พริ้นเตอร์
- วิธีที่คอมพิวเตอร์ควบคุมอุปกรณ์ได้
 - ให้อุปกรณ์เชื่อมกับหน่วยความจำค่าคงที่ค่าหนึ่ง
 - มีไว้เฉพาะกับอุปกรณ์นั้นๆ
 - เช่น คีย์บอร์ดอยู่ตำแหน่งหน่วยความจำค่าหนึ่งๆ



Flow Control

- บ่อยครั้งซีพียูทำงานหลายคำสั่ง
- มีงานซ้ำบ้าง อาจบ่อยครั้ง
- ถ้าเตรียมส่วน**โปรแกรม**ที่เรียกซ้ำบ่อยไว้ ในหน่วยคำสั่งส่วนหนึ่ง
- เมื่อต้องการใช้เพียงส่งคำสั่ง กระโดดไปตำแหน่งเริ่มต้น**โปรแกรม**
- เรียกคำสั่งกระโดดจากหน่วยความจำตำแหน่งหนึ่งไปอีกตำแหน่งว่า Jump

```
101 Load R1, 0
102 Add 1, R1
103 ...
..      // Do something with R1's value
...
156 Jump 102
```

Flow Control

- เมื่อต้องการ jump เมื่อเข้าเงื่อนไข

```
JGT R1, 0, cont      //Jump if R1 > 0
Subtract R1, 0, R1    //R1 ← (0-R1)
cont: ...
// Do something with positive R1
```

Lecture plan

- 4.1 ภาษาเครื่อง
- 4.2 ส่วนประกอบพื้นฐานของภาษาเครื่อง
- 4.3 ระบบแอสกคคอมพิวเตอร์และภาษาเครื่อง
- **4.4 ภาษาเครื่องแอสค**
- 4.5 อินพุต / เอาท์พุต
- 4.6 การเขียนโปรแกรมสำหรับเครื่องแอสค
- 4.7 ภาพรวมโปรเจค #4