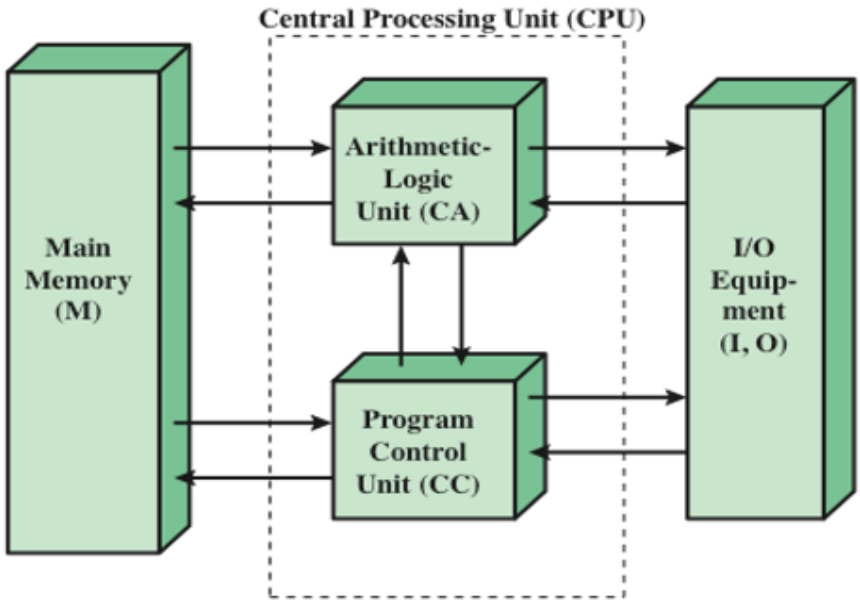
# สรุปแนวข้อสอบ Computer Architecture 59

## 1.โครงสร้าง IAS Computer (โจทย์จะให้ Block รูปภาพมาแล้วอธิบายตามภาพ)

### โครงสร้างคอมพิวเตอร์ของ John von Neumann




**หน่วยความจำหลัก ( Main Memory)** เป็น**แหล่งที่ใช้จัดเก็บได้ทั้งข้อมูลและคำสั่ง** เช่น HDD,RAM เป็นต้น

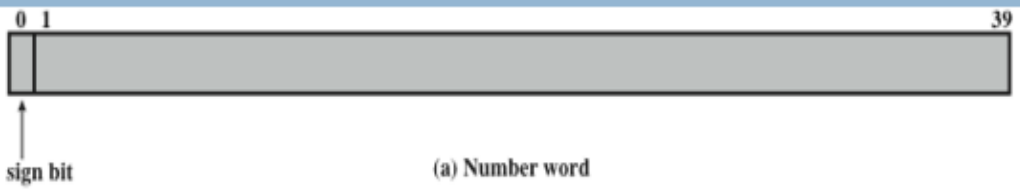
**หน่วยคำนวณและเปรียบเทียบ (Arithmetric Logic Unit : ALU)** **ใช้คำนวณคำสั่งทางคณิตศาสตร์** เช่น บวก ลบ คูณ หาร เป็นต้น

**หน่วยควบคุม (Control Unit)** เป็นหน่วยที่ใช้ในการ**ควบคุมและสั่งให้อุปกรณ์ต่างๆ ทำงาน**ตามคำสั่ง

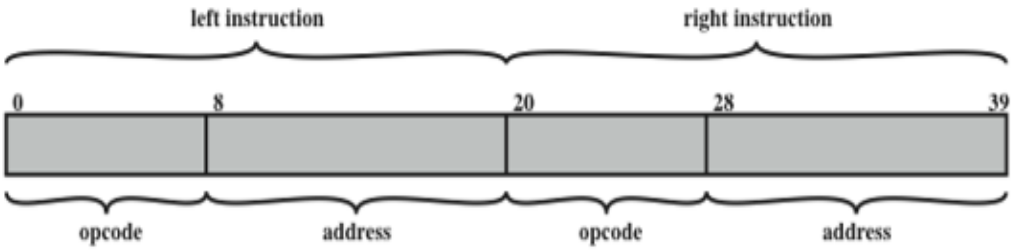
**อุปกรณ์ Input/Output** เป็นอินเตอร์เฟสเพื่อเชื่อมต่ออุปกรณ์ เช่น **อินพุตก็คือ เม้าส์ คีย์บอร์ด และ เอาต์พุตก็คือ หน้าจอแสดงผล** เป็นต้น

## 2. โครงสร้าง IAS Computer (โจทย์จะให้ Block รูปภาพมาแล้วอธิบายตามภาพ)

# IAS Memory Format




(a) Number word




(b) Instruction word

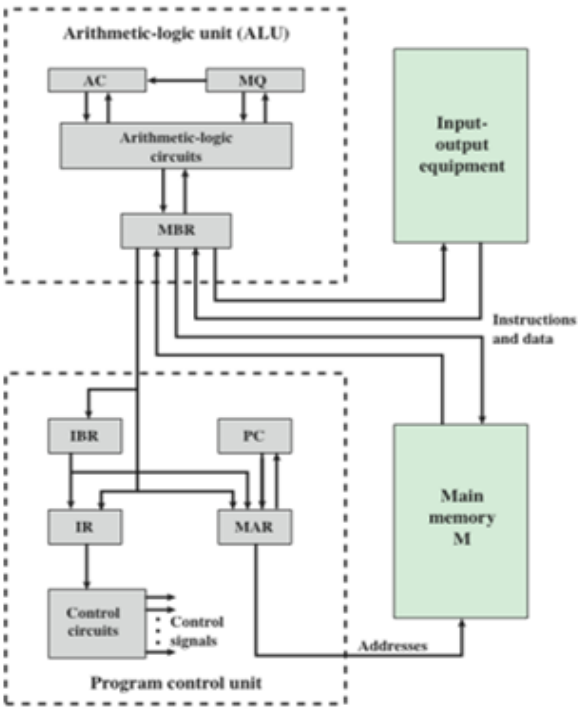
- หน่วยความจำใน IAS computer มีทั้งหมด 1000 ตำแหน่ง แต่ละตำแหน่งเก็บข้อมูล 1 word (40 bits)
- ทั้งข้อมูล (Data) และชุดคำสั่ง (Instruction) เก็บอยู่ในหน่วยความจำ

หน่วยความจำของ IAS เก็บข้อมูลได้ 1000 ตำแหน่ง หรือ 1 กิโลเวิร์ด 1 เวิร์ด = 40 บิต

คำสั่ง 1 คำสั่ง ใช้ 20 บิต ใน 20 บิต แบ่งเป็น opcode 8 บิต และ address 12 บิต

opcode บอกว่าคำสั่งนี้คือคำสั่งอะไร และ address ก็คือตำแหน่งที่อยู่ของคำสั่ง

# โครงสร้างของ IAS Computer




- **MBR** (Memory Buffer Register) เก็บ word ของ data ที่รับ/ส่ง กับอุปกรณ์ I/O และหน่วยความจำ
- **MAR** (Memory Address Register) เก็บแอดเดรสที่จะใช้รับ/ส่งข้อมูลเข้า MBR
- **IR** (Instruction Register) เก็บ opcode 8-bit ที่จะสั่งทำงาน
- **IBR** (Instruction Buffer Reigster) ที่เก็บชุดคำสั่งด้านขวาชั่วคราว
- **PC** (Program Counter) เก็บแอดเดรสของชุดคำสั่งถัดไป
- **AC** (Accumulator) และ **MQ** (Multiplier quotient) เก็บข้อมูลชั่วคราวในการคำนวณ

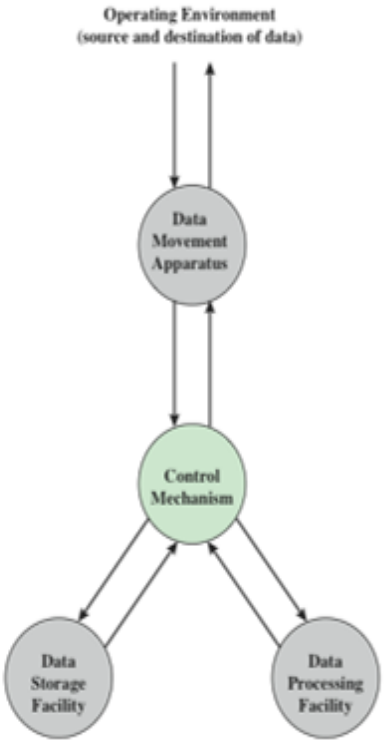
การทำงานโดยรวมดังนี้ เช่นเราใช้เครื่องคิดเลขในคอม อินพุตที่ส่งเข้ามาคอมจะมองเป็น word จะถูกเอาไปเก็บไว้ใน MBR โดย word ก็จะถูกแบ่งเป็น 2 ส่วน คือ **คำสั่ง จะถูกนำไปเก็บไว้ใน IR** และ **address จะถูกเอาไปเก็บไว้ใน MAR** โดย MBR จะอยู่ในส่วนของ ALU ทำการประมวลผลคำสั่งที่เราส่งเข้ามา โดยจะมี AC และ MQ ไว้เก็บผลลัพธ์ชั่วคราวที่คำนวณที่ได้ ส่วนการทำงานของ MAR จะอยู่ในส่วนของ Program Control Unit โดยจะควบคุมการทำงานให้กับฝั่ง ALU โดยจะมี PC เก็บ address ของคำสั่งที่จะทำต่อไป และมี IBR เก็บคำสั่งฝั่งขวา ก็คือคำสั่งอีกคำสั่งใน word ที่ถูกส่งเข้ามา เพราะ 1 word จะมีอยู่ 2 คำสั่งนั่นเอง เมื่อประมวลผลเสร็จ ก็จะส่งข้อมูลกลับไปแสดงผลที่ i/o หน้าจอคอมก็จะแสดงคำตอบให้เรากลับมานั่นเอง

## 3. concept ที่สำคัญพื้นฐานของ computer ยุคปัจจุบัน

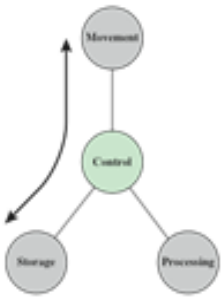
- แนวคิดนี้เรียกว่า stored program computer ซึ่ง john von Neumann เป็นผู้นำเสนอ
  - ชุดคำสั่ง (Instruction) และข้อมูล (data) จะต้องเก็บไว้ในหน่วยความจำ
  - ค่าในหน่วยความจำสามารถอ้างอิงได้จากแอดเดรสของหน่วยความจำ
  - การทำงานของอ่านชุดคำสั่งจากหน่วยความจำตามลำดับ

## 4. function of computer

**ฟังก์ชั่นการทำงานของคอมพิวเตอร์ประกอบด้วย 4 ส่วนดังนี้**

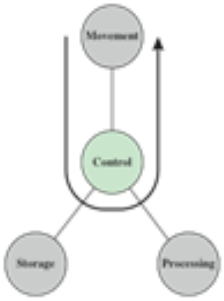



- การประมวลผลข้อมูล (Data Processing Facility)
- ส่วนเก็บบันทึกข้อมูล (Data Storage Facility)
- ส่วนการเคลื่อนย้ายข้อมูล (Data Movement Apparatus)
- ส่วนการควบคุม (Control Mechanism)

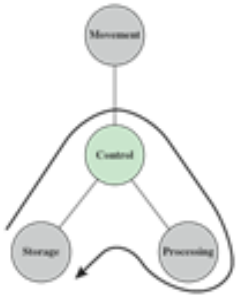



เป็นการทำงานโดยการนำข้อมูลเข้ามาเก็บไว้ในหน่วยความจำ


เป็นการทำงานโดยการนำข้อมูลจากหน่วยความมาประมวลผลแล้วส่งข้อมูลไปยัง I/O

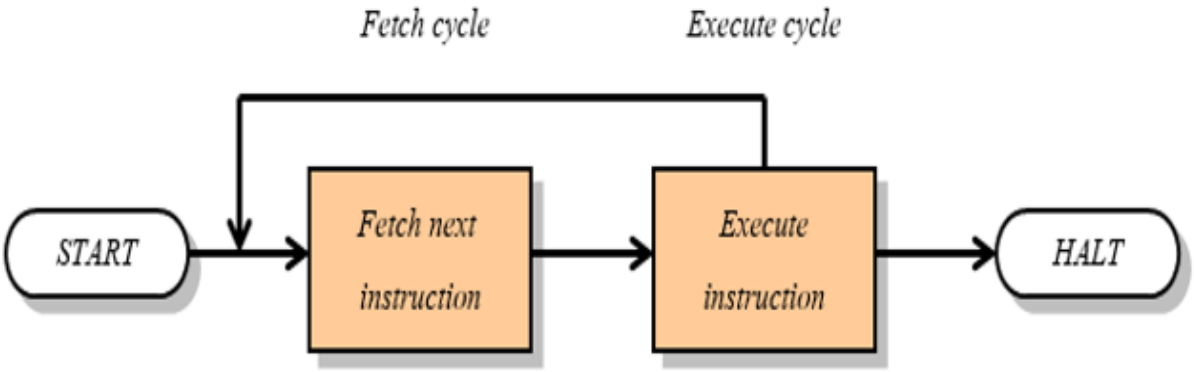



เป็นการทำงานโดยการเคลื่อนย้ายข้อมูลใน Internal หรือ External




เป็นการทำงานโดยการนำข้อมูลจากหน่วยความจำมาประมวลผลแล้วนำเข้าไปเก็บในหน่วยความจำ

## 5. การ Fetch และ Execute ข้อมูล

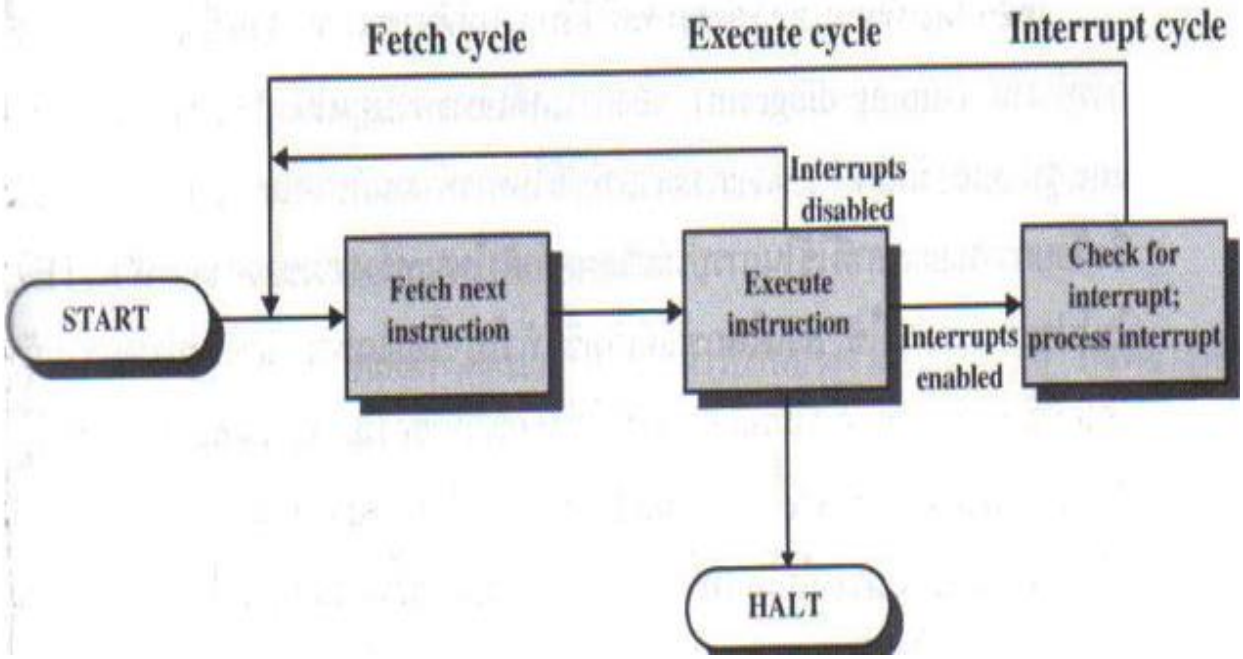



คำสั่งที่ถูกดึงมา (Fetch) จะถูกนำไปเก็บไว้ที่รีจิสเตอร์ IR คำสั่งจะประกอบด้วยบิตต่างๆ processor จะนำบิตต่างๆนี้ไปแปลความหมายและทำงาน (Execute) ตามคำสั่ง

## 6 กับ 7. interrupt วงรอบคำสั่ง และ interrupt ซ้อน

**interrupt** เป็นกลไลเพื่อเพิ่มประสิทธิภาพในการทำงานของ **processor** เมื่อเกิดการ interrupt ขึ้น **processor** จะหยุดการทำงานโปรแกรมหลัก แล้วไปทำงานตามโปรแกรมที่ interrupt เข้ามาแล้ว**เมื่อดำเนินการเสร็จ** ก็จะกลับไป**ทำงานโปรแกรมหลักดังเดิม**

อินเทอร์รัพท์และวงรอบคำสั่ง



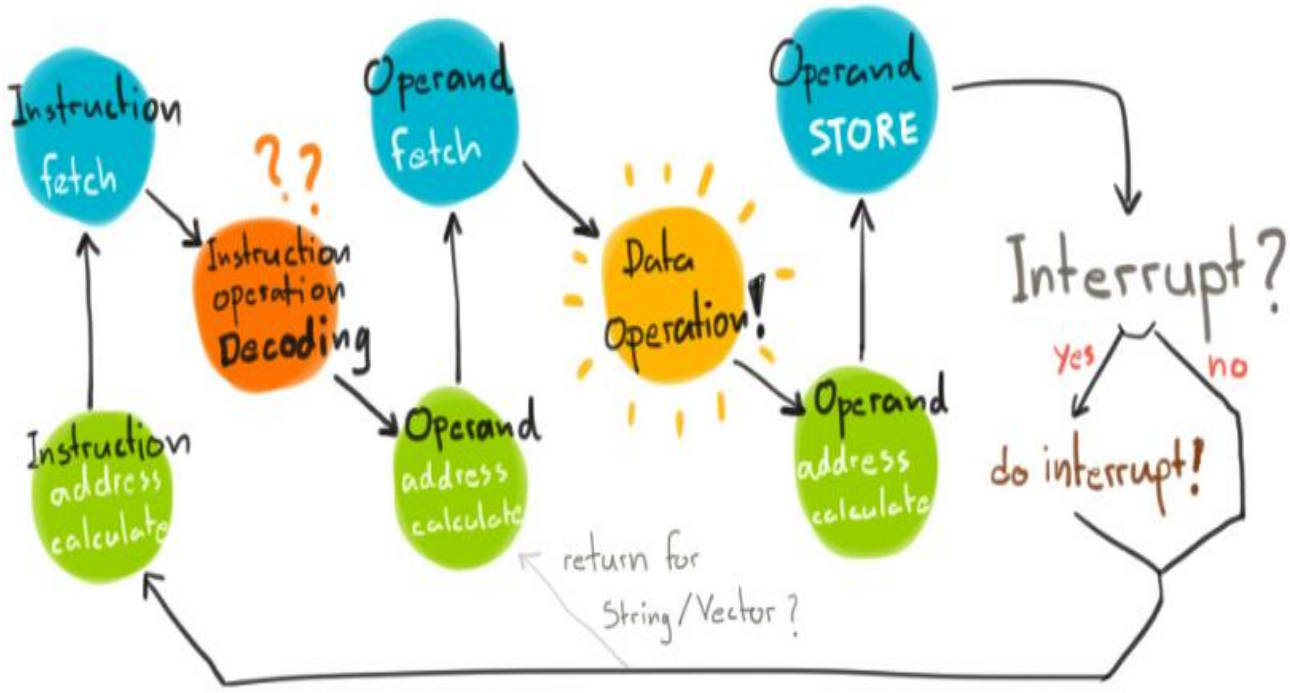

**วงรอบคำสั่ง (interrupt cycle)** processor จะตรวจสอบตรวจสอบดูว่า **มีอินเทอร์รัพท์เกิดขึ้นหรือไม่** ถ้า**ไม่มีก็จะดำเนินวงรอบการ fetch คำสั่งต่อไปเข้ามาประมวลผลปกติ** แต่**ถ้ามีสัญญาณอินเทอร์รัพท์เกิดขึ้นโปรเซสเซอร์ก็จะทำงาน**

**interrupt ซ้อน** คือการที่ processor กำลังประมวลผล interrupt ตัวใดตัวหนึ่งอยู่ ก็เกิดการ interrupt ขึ้นอีกนั่นเอง มีแนวทางแก้ปัญหาอยู่ 2 ทางคือ

1.ยกเลิกการใช้สัญญาณ **interrupt** ชั่วคราว คือการเพิกเฉย ไม่สนใจสัญญาณ **interrupt** ที่ซ้อนเข้ามา แล้วจะถูกเก็บรักษาไว้ และจะถูกตรวจสอบเมื่อ **process** กลับมาทำงานปกติ

2.กำหนดความสำคัญให้กับ **interrupt** ทุกตัว และยอมให้ **interrupt** ที่มีความสำคัญสูงกว่าทำงาน โดย **interrupt** ที่มีความสำคัญน้อยกว่าจะถูกขัดจังหวะได้เหมือนโปรแกรมทั่วไป

## 8 กับ 9.instruction cycle




**Instruction** หมายถึงคำสั่ง เช่น X = Y + 5;
**Operand** หมายถึงตำดำเนินการ ในที่นี่คือ **X** กับ **Y** นั่นเอง
**Operation** หมายถึงจะให้ทำอะไร ในที่นี้คือ + และ = (เซ็ตค่า) นะ

สมมุติให้ **คำสั่ง**ตอนนี้เก็บอยู่ที่เมมโมรี่เบอร์ 1000 , X อยู่ในเมมโมรี่เบอร์ 2000 , Y อยู่ในเมมโมรี่เบอร์ 2000 , และการ + เป็นรหัส 123 ละกันนะ
เอาล่ะ มาดู9ขั้นตอนได้แล้ว...

1. **Instruction Address Calculate**
   คำสั่งคอมพิวเตอร์เรียงกันอยู่ในเมมโมรี่ (ยังไม่ต้องถามว่ามันมายังไง ถ้าไม่มีชุดคำสั่งพวกนี้คอมก็ไม่เริ่มทำงานนะจ๊ะ) ย้ำว่าต้องเรียงกันนะ เพราะคอมพิวเตอร์จะทำงานเรียงตามนี้ วางไม่เรียงมันก็ทำงานผิดไงล่ะ ... ดังนั้นขั้นตอนแรกจึงเป็นการหาตำแหน่งของคำสั่งว่ามันอยู่ที่ไหนกันนะ
   [ได้ว่าคำสั่งเราอยู่ที่ 1000]
2. **Instruction Fetch**
   หลังจากเจอแล้วว่าคำสั่งอยู่ที่ไหน ก็ไปดึงคำสั่งออกมาสิ! ดึงออกมาในที่นี้หมายถึงดึงออกมาจากเมมโมรี่ เข้ามาในCPUซะ (สมองของคอมพิวเตอร์คือCPU ถ้าจะทำงานก็ต้องอยู่ในCPUล่ะ)
   [ดึงคำสั่งออกมาจากตำแหน่ง 1000 ซึ่งอาจจะอยู่ในรูป 0001 1110 1010 1000 1111 0011 ...]
3. **Instruction Operation 'DECODING'**
   จากขั้นตอนที่แล้วทำให้ตอนนี้เรามีคำสั่งมานอนรออยู่ในCPUแล้ว แต่คำสั่งในที่นี้อยู่ในรูปของเลขฐาน$_2$ (อีกแล้วเรอะ!) นั่นทำให้เราต้องทำการ Decode หรือถอดรหัสคำสั่งนั้นก่อนถึงจะรู้ว่าคำสั่งเนี้ยมันสั่งให้ทำอะไรกันแน่
   [จากตัวอย่างจะได้ว่า คำสั่งเมื่อกี้น่ะมันแปลว่าเราจะทำ Y + 5 แล้วเอาไปเก็บไว้ใน X นะ]
4. **Operand Address Calculate**
   หลังจากแปลความหมายได้แล้ว ส่วนใหญ่คำสั่งจะมีการยุ่งเกี่ยวกับตัวแปร เช่น X กับ Y พวกนั้น แต่ตัวแปรจะถูกเก็บอยู่ในเมมโมรี่เช่นกัน ก็เลยต้องทำคล้ายๆ ข้อที่1นั่นคือคำนวนหาที่อยู่ตัวแปรซะก่อน
   [ในคำสั่ง Y + 5 ต้องการใช้ตัวแปร Y เป็นอยู่ในตำแหน่ง 2000 ไง]
5. **Operand Fetch**
   คำนวนที่อยู่ได้แล้วก็ไปดึงตัวแปรออกมาเก็บไว้ในCPUซะ (CPUจะมีช่องเก็บของที่

เรียกว่ารีจิสเตอร์อยู่ แต่รีจิสเตอร์พวกนี้มีไม่มาก ดังนั้นต้องจัดการให้ดี)

[โอเค ไปดึงค่าที่ช่อง 2000 ออกมา สมมุติว่ามีค่า 3]

6. **Data 'Operation'**

พอได้ทั้งคำสั่งและตัวแปรที่ต้องทำงานแล้ว ก็ทำงานซะ! คำสั่งให้ + ก็บวก คำสั่งให้ – ก็ลบแบบนั้นเลย

[ในตัวอย่างให้ทำ + ก็เอาค่า 3 ที่เพิ่งดึงออกมาเมื่อกี้ไปบวกกับ 5 จะได้ 8]

7. **Operand Address Calculate**

ในขั้นตอนที่ผ่านมาทำให้เราได้คำตอบของคำสั่งนั้นแล้ว แต่คำตอบที่ได้น่ะมันยังคาอยู่ใน CPU ปัญหาคือเราต้องใช้CPUเพื่อรันคำสั่งต่อไป (อย่างที่บอกไปว่ารีจิสเตอร์มีจำกัดไงล่ะ) เลยต้องย้ายคำตอบนี้ไปเก็บไว้ในตัวแปรในเมมโมรี่เสียก่อน

[คำสั่งบอกว่าผลที่ได้ให้เอาไปใส่ตัวแปร X เลยต้องคำนวนให้ได้ก่อนว่า X คือตำแหน่ง 1000]

8. **Operand Store**

ปกติในขั้นนี้เราจะทำการ Fetch แต่มาถึงตอนนี้มันจะกลับกัน คือค่าของเรามันอยู่ใน CPUอยู่แล้ว เราจะเอาไปใส่ในเมมโมรี่ เลยเปลี่ยนจาก Fetch -> Store แทน ... เอาคำตอบไปใส่ในตำแหน่งที่ถูกต้องเป็นอันจบพิธี!

[ก็เอาคำตอบคือ 8 เมื่อกี้ไปใส่คืนที่ตำแหน่ง 1000]

9. **INTERRUPT!**

ทุกรอบที่คอมพิวเตอร์ทำงานจะมีการเช็กว่ามีความต้องการของระบบที่ขอ**ขัดจังหวะ**ซะหน่อยมั้ย ถ้ามีแล้วอนุญาตให้ทำพอดี คอมพิวเตอร์ก็จะแว็บไปทำงานนั้นแป๊ปนึง

# 10.Bus

## System Bus ประกอบด้วย **3 แบบ ดังนี้**

1.สายสัญญาณข้อมูล ทำหน้าที่ ย้ายข้อมูลระหว่างอุปกรณ์ มักมีหลายเส้น เรียกว่า ดาต้าบัส

2.สายสัญญาณตำแหน่ง ทำหน้าที่ กำหนดแหล่งที่มาของข้อมูลและแหล่งรับข้อมูล

3.สายสัญญาณควบคุม ทำหน้าที่ ควบคุมการทำงานของสายสัญญาณข้อมูลและสายสัญญาณตำแหน่งที่อยู่ การควบคุมแบ่งเป็น 2 ส่วนคือ Memory write นำข้อมูลจากสายดาต้าบัสมาบันทึกลงหน่วยความจำที่กำหนดไว้ในสัญญาณตำแหน่ง และ Memory read อ่านข้อมูลจากสายสัญญาณตำแหน่งที่อยู่เข้าสู่สายดาต้าบัส

โครงสร้างสายสัญญาณ

I/O write : ทำให้ข้อมูลที่อยู่ในดาต้าบัส บันทึกลงลงสายสัญญาณตำแหน่ง

I/O read : ทำให้เกิดการอ่านข้อมูลจากสายสัญญาณตำแหน่งเข้าสู่สายดาต้าบัส

**Transfer ACK : การอ่านหรือการบันทึกข้อมูลนั้น เสร็จเรียบร้อยแล้ว**

Bus request : อุปกรณ์ตัวที่ส่งสัญญาณออกมานั้นต้องการใช้บัส

Bus grant : อนุญาตให้ใช้บัสได้

Interrupt request : มีสัญญาณอินเทอร์รัพท์เข้ามา

Interrupt ACK : สัญญาณตอบรับอินเทอร์รัพท์

Clock : สัญญาณนาฬิกา ไม่ใช่ clock cpu แต่คือ clock bus

Reset : เริ่มกระบวนการใหม่เตรียมพร้อมการใช้งาน