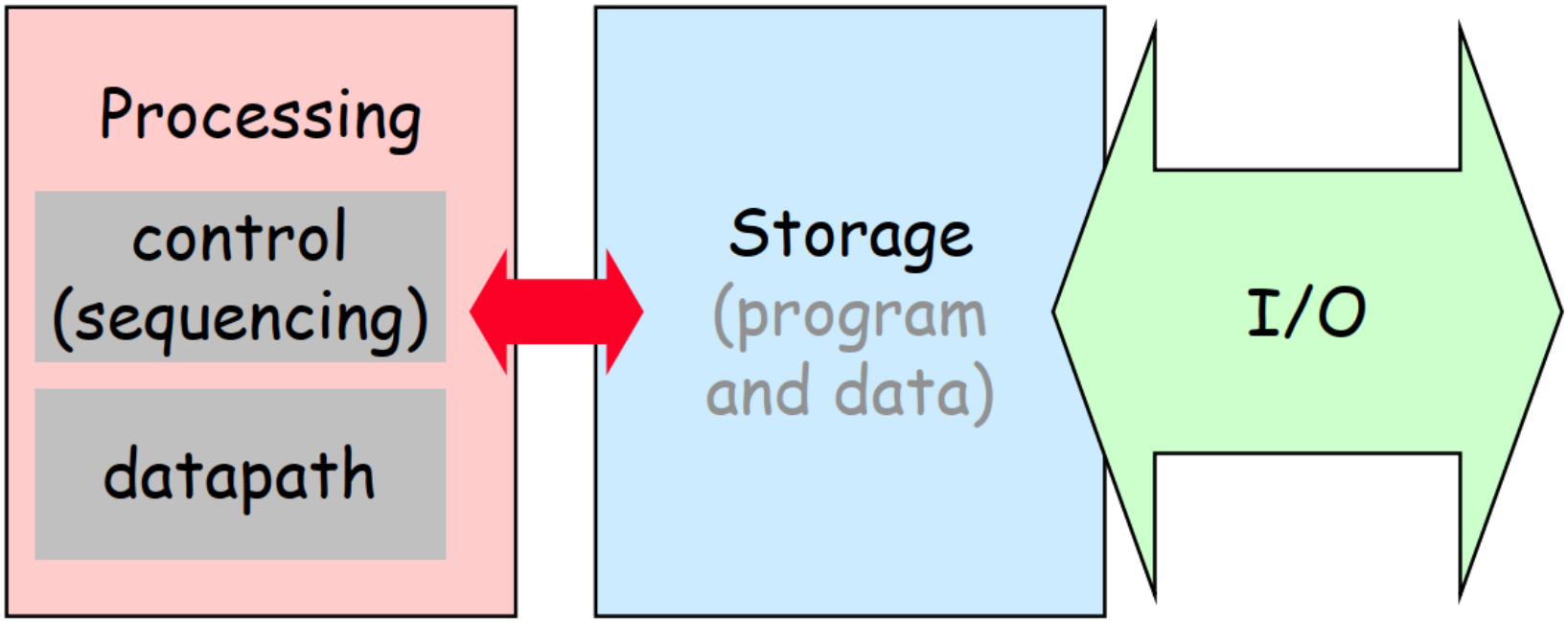
Basic Concepts of Microprocessor

หน้าที่พื้นฐานของคอมพิวเตอร์



หน้าที่พื้นฐานของคอมพิวเตอร์

- การประมวลผลข้อมูล (Data processing):
  - ข้อมูลมีได้หลากหลายรูปแบบ และมีความต้องการในการประมวลผลที่หลากหลายเช่นกัน
  - แต่มีวิธีการหรือประเภทของการประมวลผลข้อมูลพื้นฐานเพียงไม่กี่อย่างเท่านั้น
  - ไม่ว่าคอมพิวเตอร์จะได้รับข้อมูลแบบไหนมา ก็จะใช้วิธีการพื้นฐานเหล่านี้ในการจัดการกับข้อมูลให้ได้ผลลัพธ์ที่ต้องการ เช่น การคำนวณ, การจัดเรียง, การกรอง, หรือการแปลงข้อมูล

หน้าที่พื้นฐานของคอมพิวเตอร์

- การจัดเก็บข้อมูล (Data storage)
  - แม้คอมพิวเตอร์จะประมวลผลข้อมูลแบบทันทีทันใด (คือ ข้อมูลเข้ามา ประมวลผล และส่งผลลัพธ์ออกไปทันที) แต่คอมพิวเตอร์ก็ยังต้องจัดเก็บข้อมูลชั่วคราว (อย่างน้อยที่สุดคือข้อมูลที่กำลังถูกใช้งานอยู่ในขณะนั้น) ดังนั้นคอมพิวเตอร์จึงมีหน้าที่หลักในการจัดเก็บข้อมูลระยะสั้น
  - นอกจากนี้คอมพิวเตอร์ยังทำหน้าที่จัดเก็บข้อมูลระยะยาวไว้ในหน่วยความจำถาวรเพื่อเรียกใช้และอัปเดตในภายหลัง

หน้าที่พื้นฐานของคอมพิวเตอร์

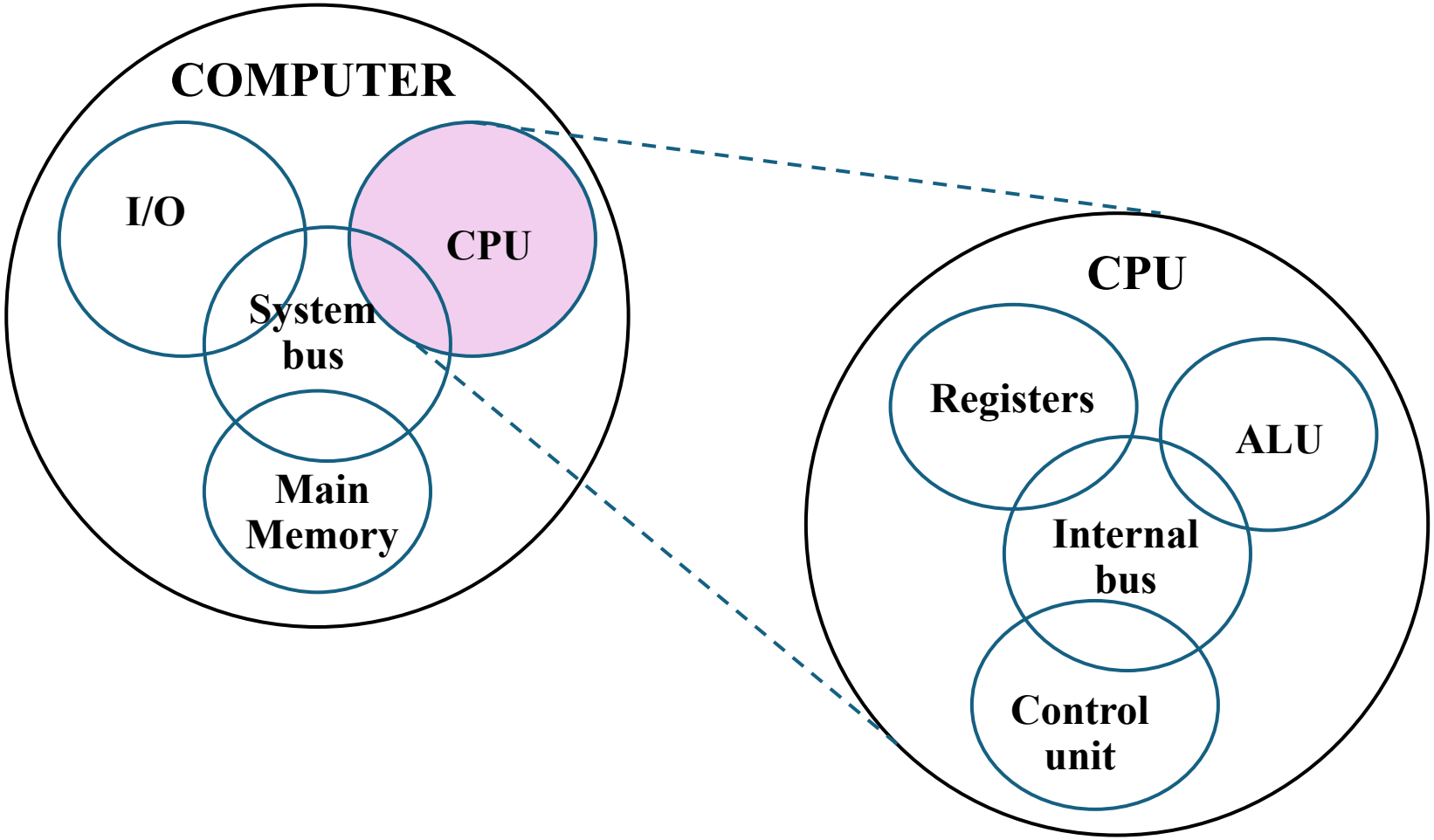
- การเคลื่อนย้ายข้อมูล (Data movement)
  - คอมพิวเตอร์ประกอบด้วยอุปกรณ์ที่เป็นแหล่งกำเนิดข้อมูล (source) หรือปลายทางของข้อมูล (destination)
  - ข้อมูลถูกรับมาจากหรือส่งไปยังอุปกรณ์ที่เชื่อมต่อโดยตรงกับคอมพิวเตอร์ เรียกว่า การนำเข้า-ส่งออก (Input-Output หรือ I/O) และอุปกรณ์นั้นจะเรียกว่า อุปกรณ์ต่อพ่วง (peripheral)
  - เมื่อข้อมูลถูกย้ายในระยะทางที่ไกลขึ้น ไปยังหรือมาจากอุปกรณ์ระยะไกล กระบวนการนี้เรียกว่า การสื่อสารข้อมูล (data communications) และเรียกว่าอุปกรณ์เครือข่าย (Computer network)
  - หน้าที่นี้ทำให้คอมพิวเตอร์สามารถแลกเปลี่ยนข้อมูลกับโลกภายนอกได้ ไม่ว่าจะเป็นอุปกรณ์ที่อยู่ใกล้ๆ หรือเครือข่ายที่อยู่ไกลออกไป

หน้าที่พื้นฐานของคอมพิวเตอร์

- การควบคุม (Control)
  - ภายในคอมพิวเตอร์ มีหน่วยควบคุม (control unit) ทำหน้าที่จัดการทรัพยากรของคอมพิวเตอร์และประสานงานการทำงานของส่วนประกอบต่างๆ ของคอมพิวเตอร์ตามคำสั่งที่ได้รับ
  - หน้าที่นี้เปรียบเสมือนผู้จัดการที่คอยสั่งการและดูแลให้ทุกส่วนของคอมพิวเตอร์ทำงานร่วมกันอย่างถูกต้องตามโปรแกรม เพื่อให้บรรลุเป้าหมายที่ต้องการ

โครงสร้างพื้นฐานของคอมพิวเตอร์แบบ Single-processor

Top-Level structure



หน่วยประมวลผลกลาง (Central Processing Unit - CPU)

- เป็นส่วนที่ควบคุมการทำงานทั้งหมดของคอมพิวเตอร์ทำหน้าที่ประมวลผลข้อมูลบ่อยครั้งมักเรียกสั้นๆ ว่า "โปรเซสเซอร์" (Processor)
- เปรียบเสมือนสมองของคอมพิวเตอร์ที่คอยคิดคำนวณและสั่งการ

หน่วยความจำหลัก (Main memory)

- ทำหน้าที่จัดเก็บข้อมูลเป็นพื้นที่ที่ CPU ใช้ในการเข้าถึงข้อมูลและคำสั่งที่กำลังทำงานอยู่หรือจำเป็นต้องใช้ในทันทีทันใด
- ข้อมูลในหน่วยความจำหลักนี้จะหายไปเมื่อปิดเครื่อง (มักเป็น RAM)

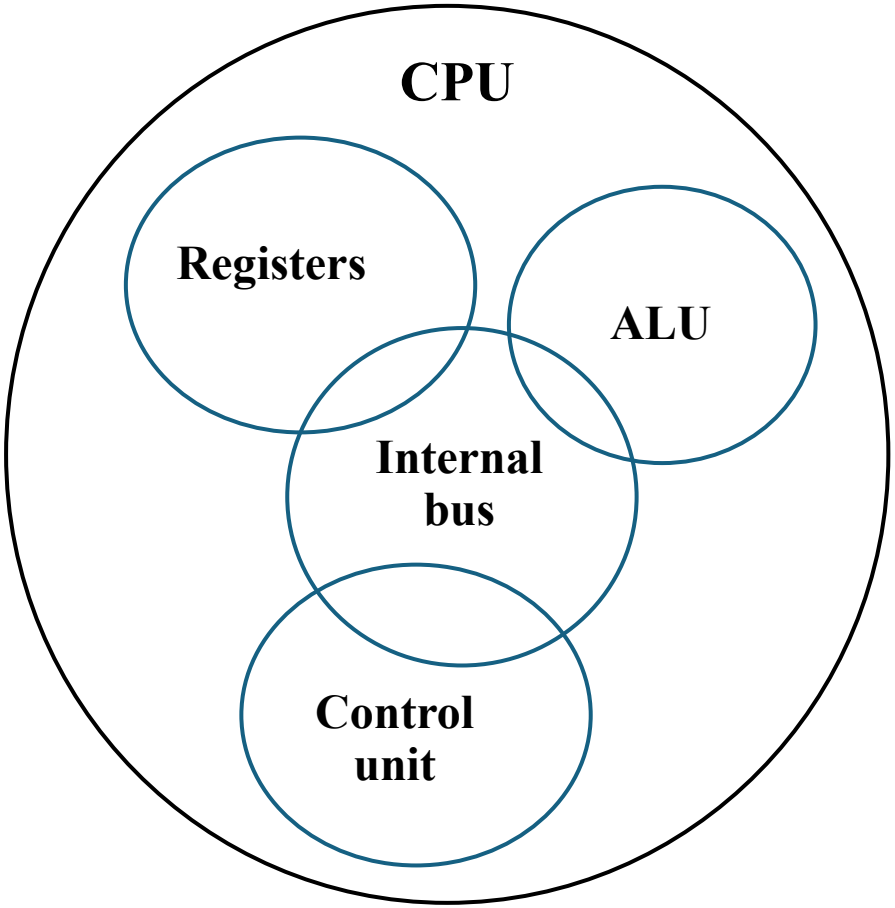
หน่วยนำเข้า/ส่งออก (I/O - Input/Output)

- ทำหน้าที่เคลื่อนย้ายข้อมูลระหว่างคอมพิวเตอร์และสภาพแวดล้อมภายนอก
- หมายความรวมถึงการที่คอมพิวเตอร์รับข้อมูลเข้ามา (Input) เช่น จากคีย์บอร์ด เมาส์ หรือสแกนเนอร์ และส่งข้อมูลออกไป (Output) เช่น ไปยังจอภาพ เครื่องพิมพ์ หรือลำโพง
- เป็นส่วนที่ทำให้คอมพิวเตอร์สามารถโต้ตอบกับผู้ใช้และอุปกรณ์ภายนอกได้

การเชื่อมต่อระบบ (System interconnection)

- เป็นกลไกที่จัดให้มีการสื่อสารระหว่าง CPU, หน่วยความจำหลัก (main memory) และหน่วยนำเข้า/ส่งออก (I/O)
- การเชื่อมต่อระบบมักหมายถึง บัสต่าง ๆ ของระบบ (system bus)
  - บัสประกอบด้วยสายนำไฟฟ้าหลายเส้นที่เชื่อมต่ออยู่กับระบบเปรียบเสมือนถนนหลักที่ข้อมูลและคำสั่งใช้เดินทางไปมาระหว่างส่วนประกอบต่าง ๆ ภายในคอมพิวเตอร์ ทำให้ทุกส่วนสามารถทำงานร่วมกันได้อย่างราบรื่น

โครงสร้างพื้นฐานของ CPU



หน่วยควบคุม (Control unit)

- ทำหน้าที่ควบคุมการทำงานของ CPU และด้วยเหตุนี้จึงควบคุมการทำงานของคอมพิวเตอร์ทั้งหมด
- เปรียบเสมือน "ผู้จัดการ" ที่คอยอ่านและตีความคำสั่ง จากนั้นก็สั่งการให้ส่วนประกอบอื่น ๆ ของ CPU ทำงานตามที่ต้องการ เพื่อให้โปรแกรมดำเนินไปอย่างถูกต้อง

หน่วยคำนวณและตรรกะ

- ชื่ออังกฤษ: Arithmetic and logic unit - ALU
- ทำหน้าที่ควบคุมการทำงานของ CPU และด้วยเหตุนี้จึงควบคุมการทำงานของคอมพิวเตอร์ทั้งหมด
- เปรียบเสมือน "ผู้จัดการ" ที่คอยอ่านและตีความคำสั่ง จากนั้นก็สั่งการให้ส่วนประกอบอื่น ๆ ของ CPU ทำงานตามที่ต้องการ เพื่อให้โปรแกรมดำเนินไปอย่างถูกต้อง

รีจิสเตอร์ (Registers)

- ทำหน้าที่จัดเก็บข้อมูลภายใน CPU
- เป็นหน่วยความจำขนาดเล็กที่มีความเร็วสูงมากที่อยู่ภายในตัว CPU โดยตรง
- ใช้สำหรับเก็บข้อมูลที่ CPU กำลังประมวลผลอยู่ชั่วคราว หรือเก็บผลลัพธ์จากการคำนวณที่ต้องใช้ในทันทีเพื่อให้การเข้าถึงข้อมูลเป็นไปอย่างรวดเร็วที่สุด

การเชื่อมต่อภายใน CPU

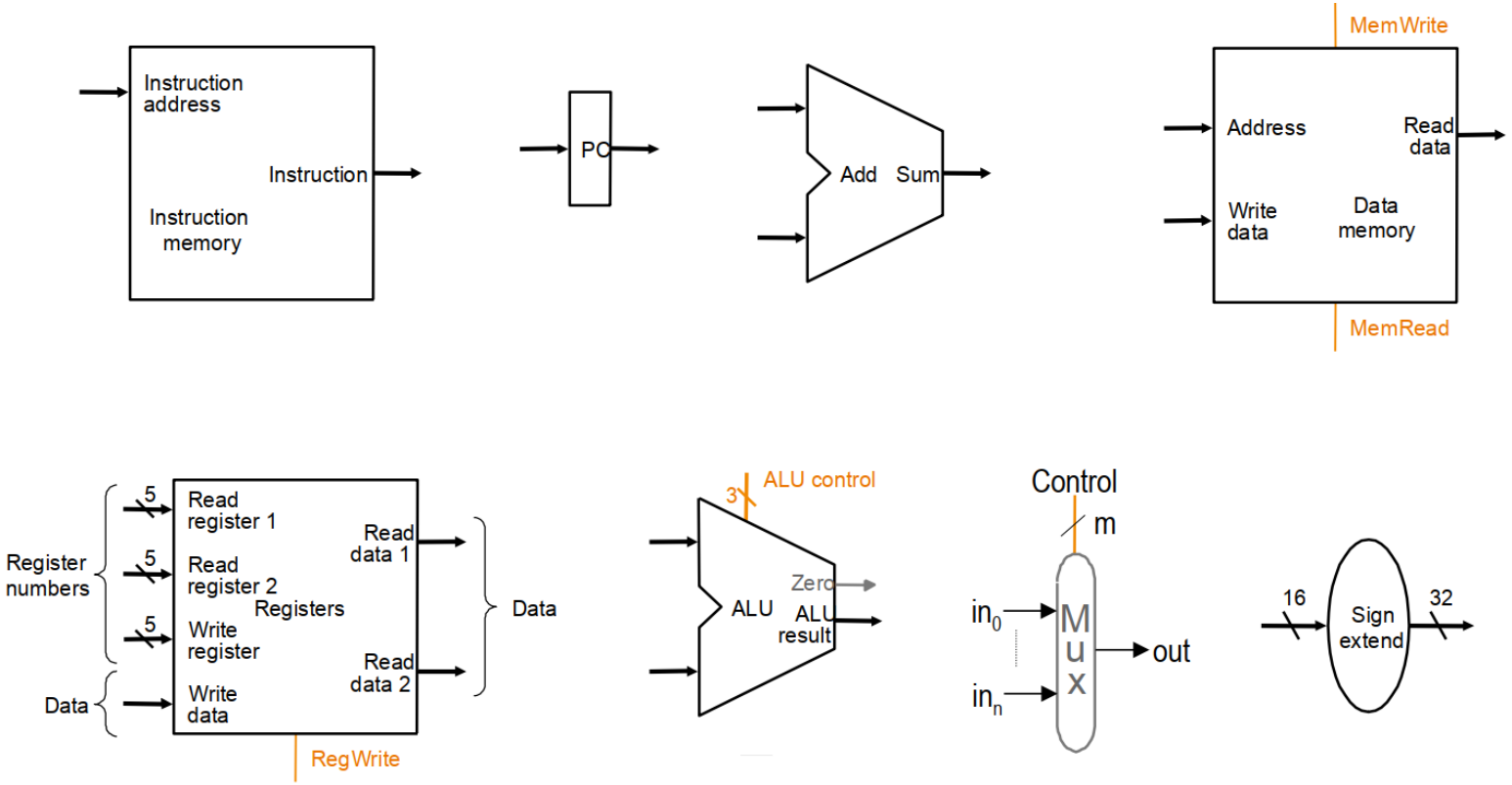
- ชื่ออังกฤษ CPU interconnection
- เป็นกลไกบางอย่างที่จัดให้มีการสื่อสารระหว่างหน่วยควบคุม (control unit), หน่วยคำนวณและตรรกะ (ALU) และรีจิสเตอร์ (registers)
- ทำหน้าที่เป็นเส้นทางในการส่งข้อมูลและคำสั่งระหว่างส่วนประกอบย่อย ๆ เหล่านี้ภายใน CPU เพื่อให้สามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพ

สถาปัตยกรรมของ CPU

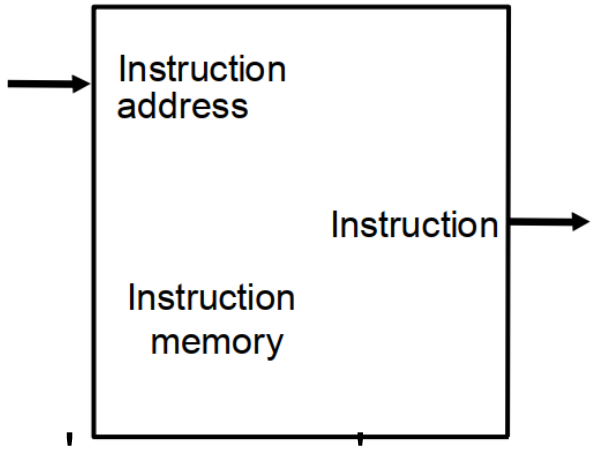
องค์ประกอบพื้นฐานภายใน CPU

ตัวอย่างสถาปัตยกรรมคอมพิวเตอร์แบบ MIPS

(Microprocessor without Interlocked Pipeline Stages)

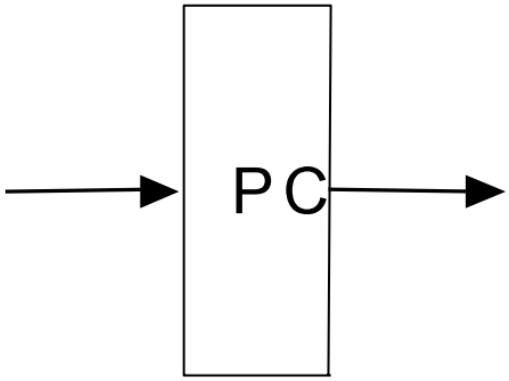


1. Instruction Memory (หน่วยความจำคำสั่ง)



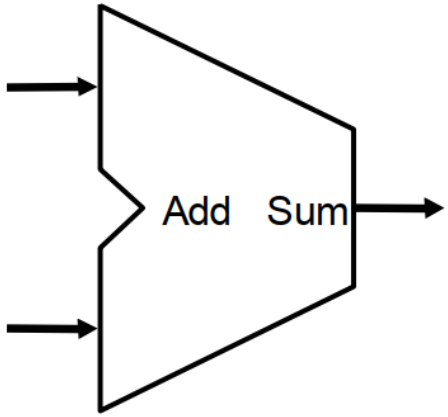
- Input: Instruction address (ที่อยู่ของคำสั่ง) – รับที่อยู่ของคำสั่งที่ต้องการดึง
- Output: Instruction (คำสั่ง) – ส่งออกคำสั่งที่อยู่ที่ที่อยู่ดังกล่าว
- บล็อกนี้ทำหน้าที่เก็บโปรแกรมที่ซีพียูจะรัน เมื่อ CPU ต้องการรันคำสั่งใด ก็จะส่งที่อยู่ของคำสั่งนั้นเข้ามา แล้วหน่วยความจำคำสั่งก็จะส่งคำสั่ง (ในรูปแบบโค้ดเลขฐานสอง) ออกไป

2. Program Counter (PC) (ตัวนับโปรแกรม)



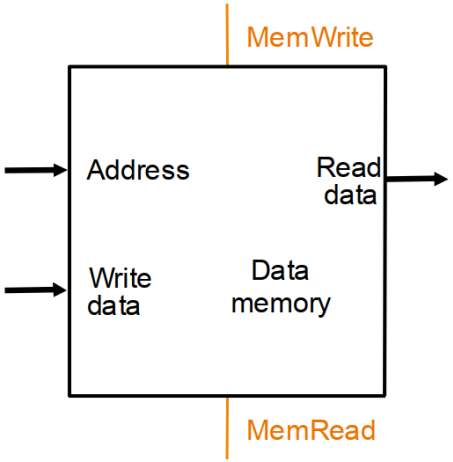
- บล็อกเล็กๆ ที่มีลูกศรเข้าและออก แสดงถึง Program Counter (PC)
- เป็นรีจิสเตอร์พิเศษที่เก็บที่อยู่ของคำสั่งถัดไปที่จะถูกประมวลผล
- PC จะถูกอัปเดตค่าเสมอเพื่อชี้ไปยังคำสั่งถัดไปในการดำเนินโปรแกรม

3. Adder (Add/Sum)
(หน่วยบวก)



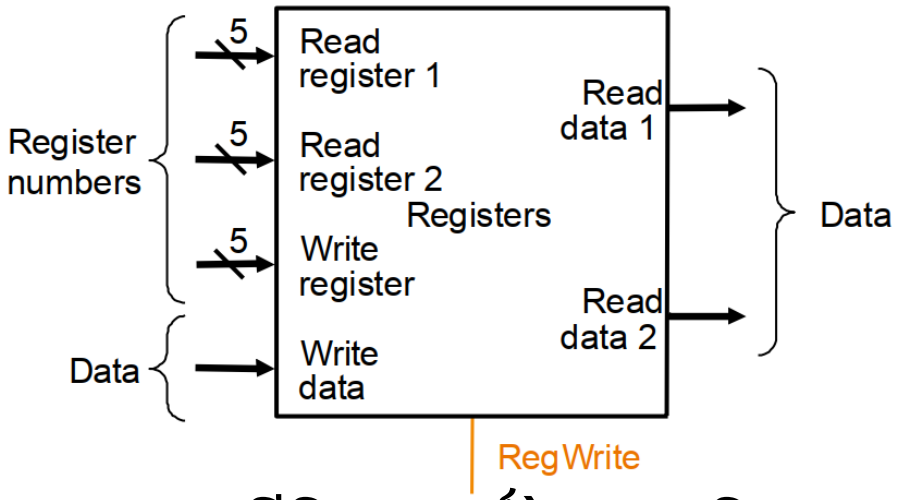
- Input: สองค่าที่ต้องการบวก (ในบริบทนี้ อาจจะเป็นค่า PC ปัจจุบัน + 4 เพื่อชี้ไปยังคำสั่งถัดไป หรือใช้ในการคำนวณที่อยู่สำหรับการกระโดด (branch/jump))
- Output: Sum (ผลรวม)
- บล็อกนี้ทำหน้าที่บวกเลขสองจำนวน มักใช้ในการคำนวณที่อยู่ของคำสั่งถัดไป หรือที่อยู่ข้อมูลในหน่วยความจำ

4. Data Memory (หน่วยความจำข้อมูล)



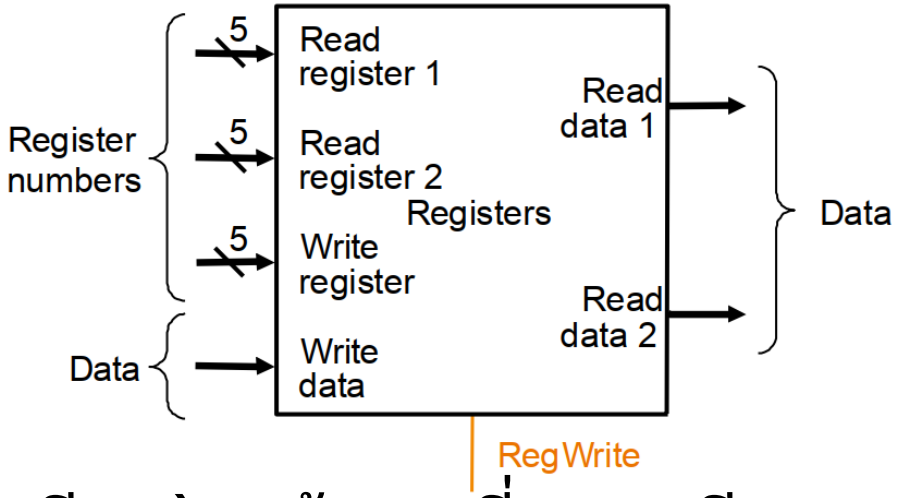
- Input: Address (ที่อยู่) - ที่อยู่ของข้อมูลที่ต้องการอ่านหรือเขียน, Write data (ข้อมูลที่ต้องการเขียน)
- Control Signals
  - MemWrite: สัญญาณควบคุมการเขียนข้อมูล (ถ้าเป็น 1 คือเขียน, ถ้าเป็น 0 คือไม่อะไรเลย)
  - MemRead: สัญญาณควบคุมการอ่านข้อมูล (ถ้าเป็น 1 คืออ่าน, ถ้าเป็น 0 คือไม่อะไรเลย)Output: Read data (ข้อมูลที่อ่านได้)
- บล็อกนี้ใช้สำหรับเก็บข้อมูลที่โปรแกรมใช้งานแยกจากคำสั่ง มีหน้าที่อ่านและเขียนข้อมูลไปยังที่อยู่หน่วยความจำที่ระบุ

5. Registers (กลุ่มรีจิสเตอร์)



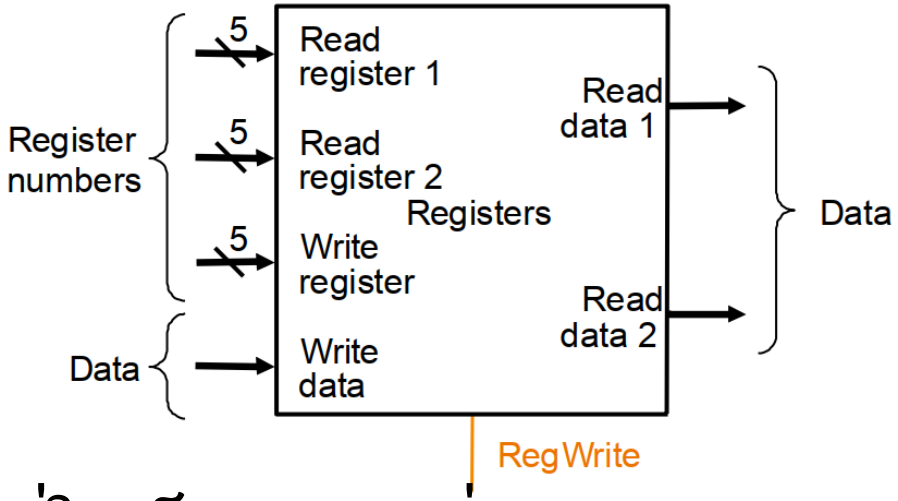
- Input: Register numbers (หมายเลขรีจิสเตอร์) - 5 บิต สำหรับแต่ละรีจิสเตอร์ที่ต้องการอ่านหรือเขียน (เนื่องจาก MIPS มี 32 รีจิสเตอร์, 2 5 =32)
  - Read register 1: หมายเลขของรีจิสเตอร์ตัวแรกที่ต้องการอ่าน
  - Read register 2: หมายเลขของรีจิสเตอร์ตัวที่สองที่ต้องการอ่าน
  - Write register: หมายเลขของรีจิสเตอร์ที่ต้องการเขียนข้อมูล

5. Registers (ต่อ)



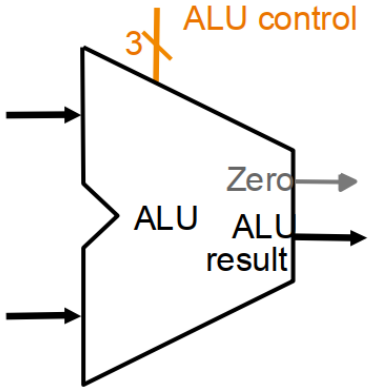
- Input: Write data (ข้อมูลที่จะเขียน) – ข้อมูลที่จะถูกเขียนลงในรีจิสเตอร์ที่ระบุ
- Control Signal: RegWrite (สัญญาณควบคุมการเขียนรีจิสเตอร์) - ถ้าเป็น 1 คือเขียน, ถ้าเป็น 0 คือไม่ทำอะไรเลย
- Output: Read data 1, Read data 2 (ข้อมูลที่อ่านได้จากรีจิสเตอร์ 1 และ 2)
- Output: Data (เส้นทางข้อมูลทั่วไป)

5. Registers (ต่อ)



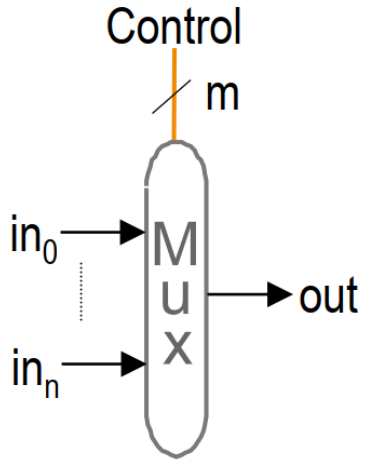
- บล็อกนี้เป็นชุดของรีจิสเตอร์ที่ใช้เก็บข้อมูลชั่วคราวภายใน CPU โดยตรง
- รีจิสเตอร์เหล่านี้มีความเร็วในการเข้าถึงสูงมาก
  - ใช้สำหรับเก็บค่าตัวแปร, ผลลัพธ์ระหว่างการคำนวณ หรือที่อยู่ต่างๆ

6. ALU (Arithmetic Logic Unit) (หน่วยคำนวณและตรรกะ)



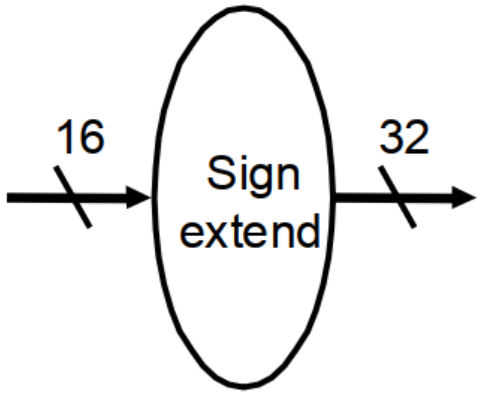
- Input: ค่าของ Data สองตัวที่จะนำมาคำนวณ
- Control Signal: ALU control (สัญญาณควบคุมการทำงานของ ALU)
  - กำหนดว่า ALU จะทำการบวก ลบ AND OR ฯลฯ
- Output: ALU result (ผลลัพธ์จาก ALU)
  - ผลลัพธ์จากการดำเนินการOutput: Zero (สัญญาณ Zero) - เป็น 1 ถ้าผลลัพธ์เป็นศูนย์, เป็น 0 ถ้าไม่เป็นศูนย์
- ALU ทำหน้าที่ดำเนินการทางคณิตศาสตร์ (เช่น บวก, ลบ) และตรรกะ (เช่น AND, OR, XOR) ตามที่สัญญาณควบคุมกำหนด

7. Mux (Multiplexer)
(มัลติเพล็กเซอร์)



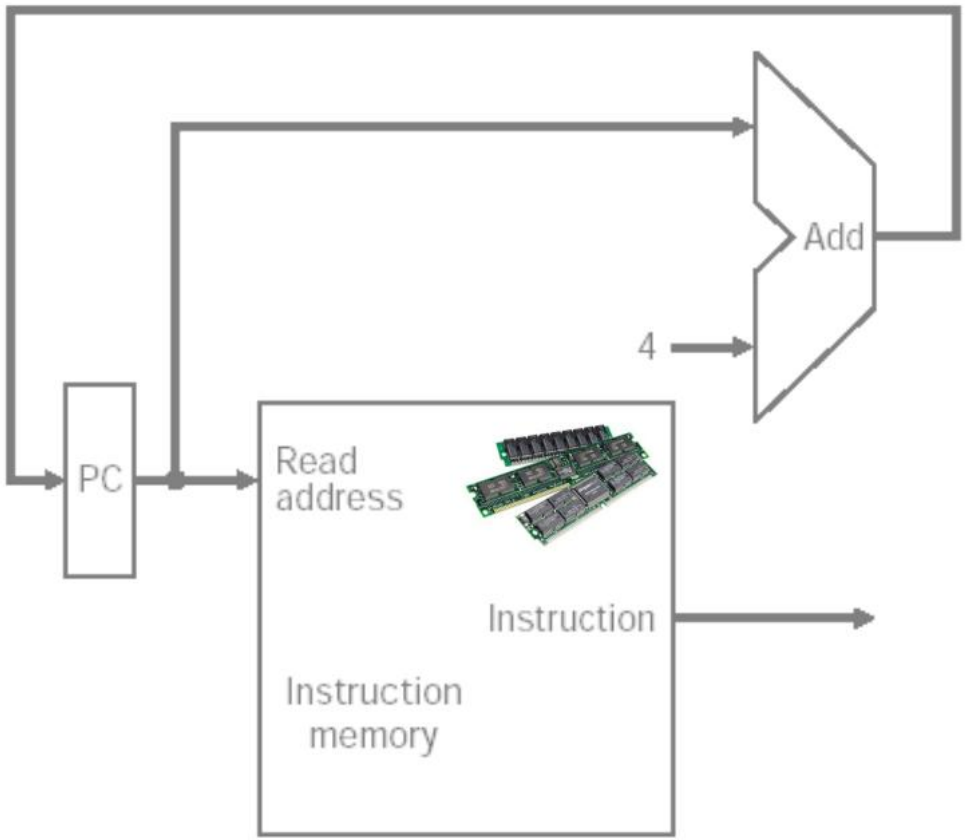
- Input: in0, in1, ... inn (อินพุตหลายค่า)
  - ในภาพมี 2 อินพุตคือ in0 และ in1 (และมีจุดแสดงว่าอาจมีมากกว่านั้น)
- Control Signal: m (สัญญาณเลือก)
  - กำหนดว่าจะเลือกอินพุตตัวไหนออกไป
- Output: out (เอาต์พุตที่เลือก)
- มัลติเพล็กเซอร์ทำหน้าที่เลือกข้อมูลจากอินพุตหลายช่องทางตามสัญญาณควบคุม m แล้วส่งออกไปยังเอาต์พุตเดียว เปรียบเสมือนสวิตช์เลือกข้อมูล

8. Sign Extend (หน่วยขยายเครื่องหมาย)

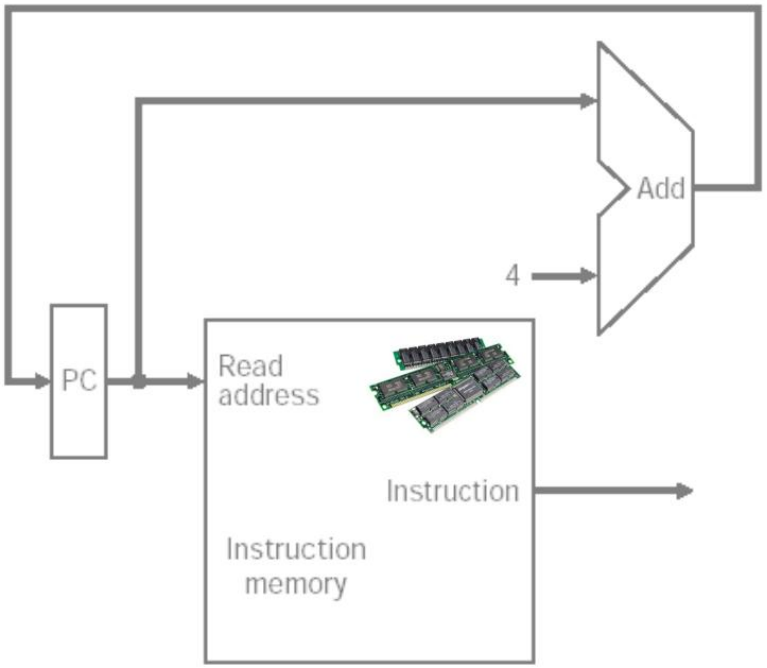


- Input: ค่าเลข 16 บิต (แสดงด้วย 16 ที่ลูกศรขาเข้า)
- Output: ค่าเลข 32 บิต (แสดงด้วย 32 ที่ลูกศรขาออก)
- บล็อกนี้ทำหน้าที่ขยายขนาดของตัวเลขจาก 16 บิต (ซึ่งมักใช้สำหรับค่าคงที่ขนาดเล็กหรือ offsets ในคำสั่ง) ให้เป็น 32 บิต
  - โดยการคัดลอกบิตเครื่องหมาย (most significant bit) ไปเติมในบิตที่เพิ่มเข้ามา เพื่อรักษามูลค่าของเลขจำนวนเต็มให้ถูกต้องทั้งบวกและลบ

Fetching Instructions (no branching)



1. Program Counter (PC)

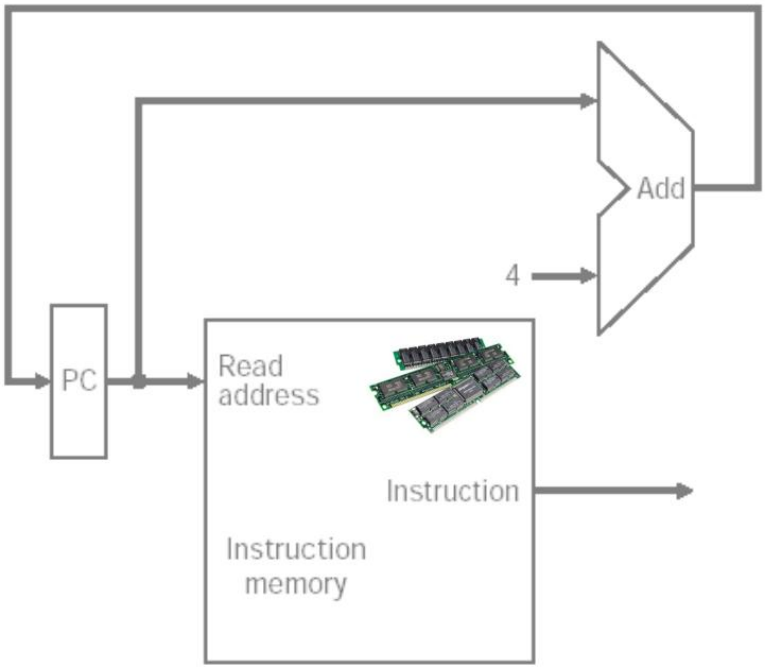


- PC (Program Counter) เป็นรีจิสเตอร์ที่เก็บ "ที่อยู่" ของคำสั่งถัดไปที่ CPU ต้องการจะดึงและประมวลผล
- ลูกศรที่ออกจาก PC แสดงว่าค่าที่อยู่ปัจจุบันใน PC กำลังถูกส่งต่อไปยังสองส่วน

1. ส่งไปยังหน่วยความจำคำสั่ง (Instruction memory) เพื่อใช้เป็นที่อยู่ในการอ่านคำสั่ง

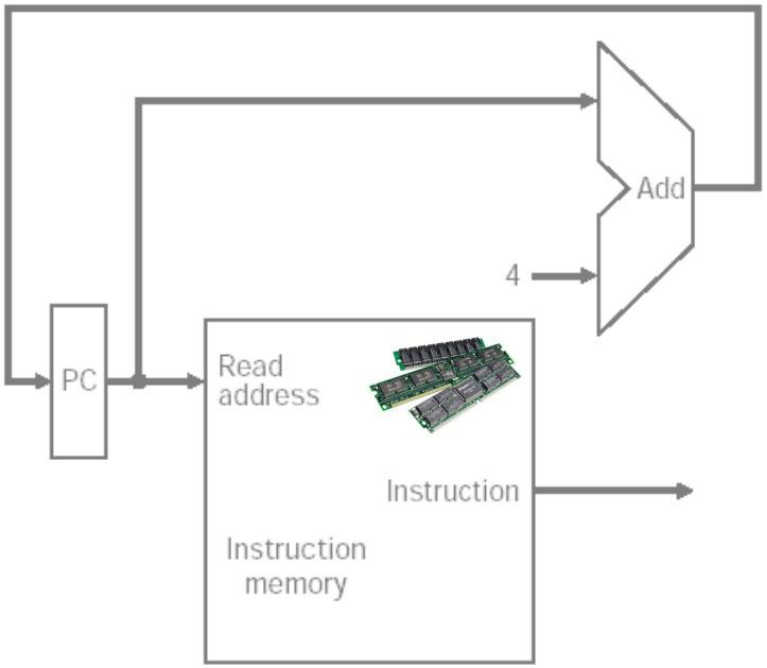
2. ส่งไปยังหน่วยบวก (Add) เพื่อใช้ในการคำนวณที่อยู่ของคำสั่งถัดไป

2. Instruction Memory (หน่วยความจำคำสั่ง)



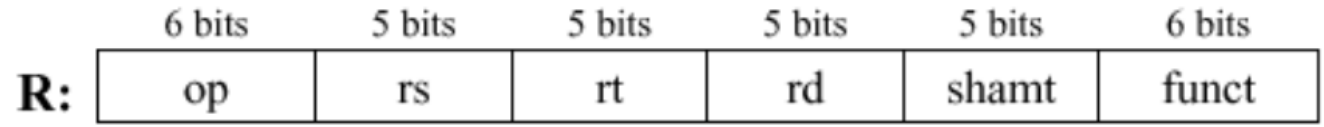
- ลูกศรจาก PC ชี้มาที่ช่อง Read address ของบล็อก Instruction memory
- บล็อก Instruction memory เป็นส่วนที่เก็บคำสั่ง (โปรแกรม) ทั้งหมดของคอมพิวเตอร์
- เมื่อได้รับที่อยู่จาก PC หน่วยความจำคำสั่งจะค้นหาและดึงคำสั่งที่อยู่ในที่อยู่นั้น
- จากนั้น คำสั่งจะถูกส่งออกไปทางลูกศร Instruction ไปยังส่วนอื่นๆ ของ CPU เพื่อทำการถอดรหัสและประมวลผลต่อไป

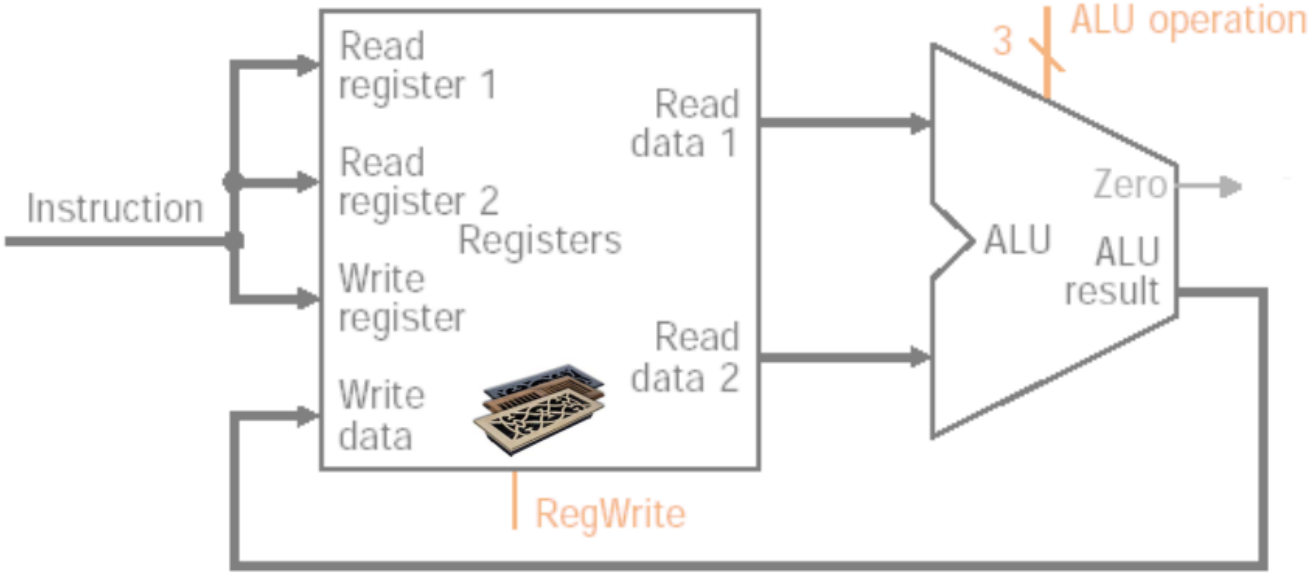
3. Adder (หน่วยบวก)



- ค่าที่อยู่ปัจจุบันจาก PC ถูกส่งไปยังอินพุตด้านบนของบล็อก Add
- อินพุตอีกด้านหนึ่งของบล็อก Add คือค่าคงที่ "4" (แสดงด้วยลูกศรที่มีเลข 4)
  - ค่า "4" นี้สำคัญมาก เนื่องจากในสถาปัตยกรรมคอมพิวเตอร์ส่วนใหญ่นั้น คำสั่งแต่ละคำสั่งมีขนาด 4 ไบต์ ดังนั้น การเพิ่ม 4 ให้กับที่อยู่ปัจจุบันจะทำให้ได้ที่อยู่ของคำสั่งถัดไป
  - บล็อก Add จะทำการบวกค่าที่อยู่จาก PC กับ 4
- ผลลัพธ์ของการบวกนี้ ซึ่งก็คือที่อยู่ของคำสั่งถัดไป จะถูกส่งกลับไปอัปเดตค่าใน PC สำหรับรอบการดึงคำสั่งครั้งหน้า

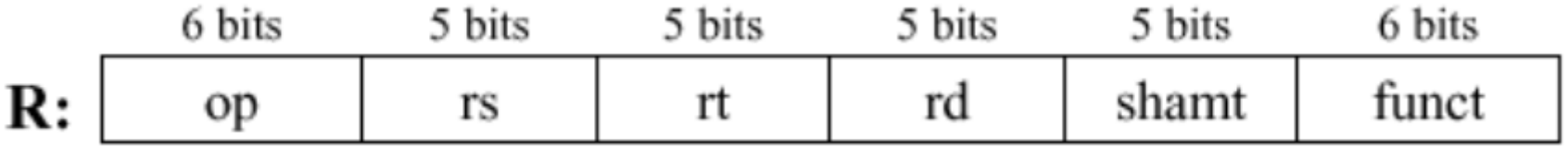
ตัวอย่างคำสั่ง R-Type





Consider: r1 = r2 - r3

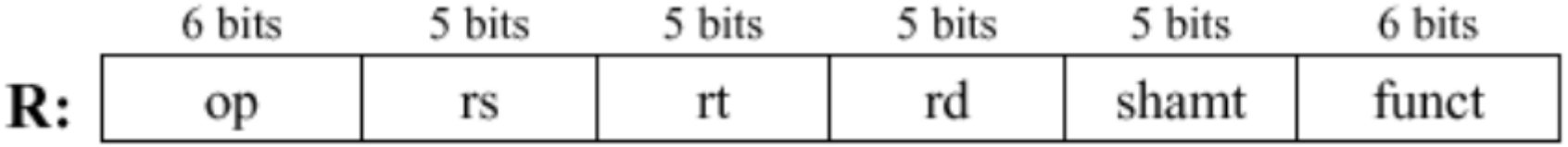
โครงสร้างคำสั่ง R-Type



คำสั่ง R-Type ใน MIPS มีขนาด 32 บิต แบ่งออกเป็นฟิลด์ต่างๆ ดังนี้

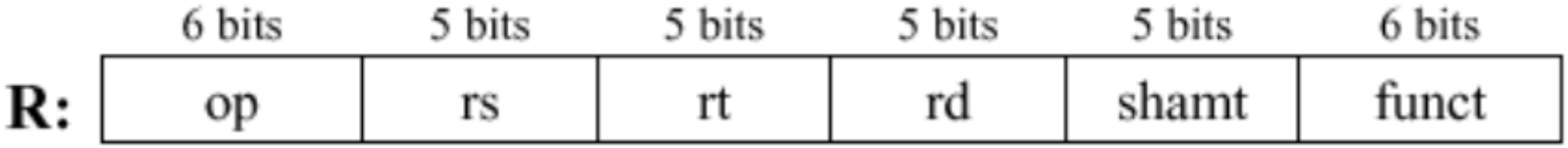
- op (opcode - 6 bits)
  - รหัสการทำงานหลักของคำสั่ง สำหรับ R-Type คำสั่งนี้มักจะเป็น 000000 เพื่อบ่งชี้ว่าเป็นคำสั่งประเภท R-Type และรายละเอียดการทำงานจะอยู่ในฟิลด์ funct
- rs (source register 1 - 5 bits)
  - หมายเลขของรีจิสเตอร์แหล่งที่มาตัวแรก (source register) ที่ข้อมูลจะถูกอ่านออกมาเพื่อนำไปประมวลผล
- rt (source register 2 - 5 bits)
  - หมายเลขของรีจิสเตอร์แหล่งที่มาตัวที่สอง (source register) ที่ข้อมูลจะถูกอ่านออกมาเพื่อนำไปประมวลผล

โครงสร้างคำสั่ง R-Type (ต่อ)



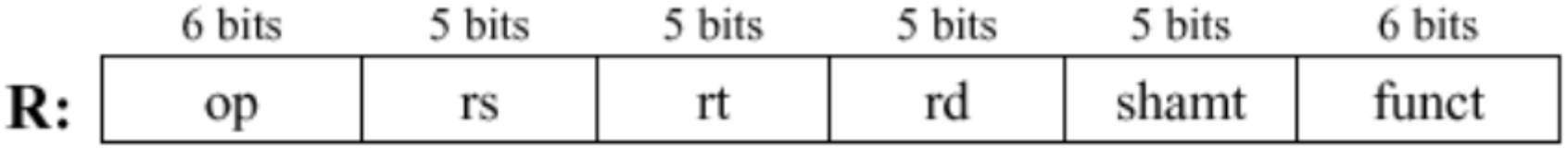
- rd (destination register - 5 bits)
  - หมายเลขของรีจิสเตอร์ปลายทาง (destination register) ที่ผลลัพธ์จากการประมวลผลจะถูกเขียนกลับเข้าไป
- shamt (shift amount - 5 bits)
  - จำนวนบิตที่จะเลื่อน (shift amount) ใช้สำหรับคำสั่งเลื่อนบิต (shift operations) ถ้าไม่ใช่คำสั่งเลื่อนบิต ค่านี้จะเป็น 0
- funct (function - 6 bits)
  - รหัสฟังก์ชันที่ระบุการทำงานที่แท้จริงของคำสั่ง R-Type เช่น การบวก การลบ การ AND การ OR เป็นต้น

การทำงานของคำสั่ง R-Type



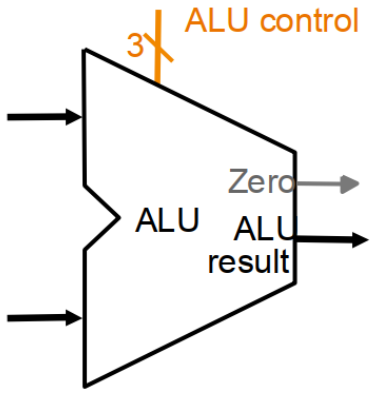
- Instruction (คำสั่ง)
  - คำสั่ง R-Type (เป็นคำสั่งขนาด 32 บิต) ที่ดึงมาจากหน่วยความจำคำสั่งจะถูกป้อนเข้าสู่ระบบ
  - ฟิลด์ rs, rt, และ rd จากคำสั่งนี้ จะถูกแยกออกมาเพื่อใช้เป็นหมายเลขรีจิสเตอร์ในการอ่านและเขียน

การทำงานของคำสั่ง R-Type



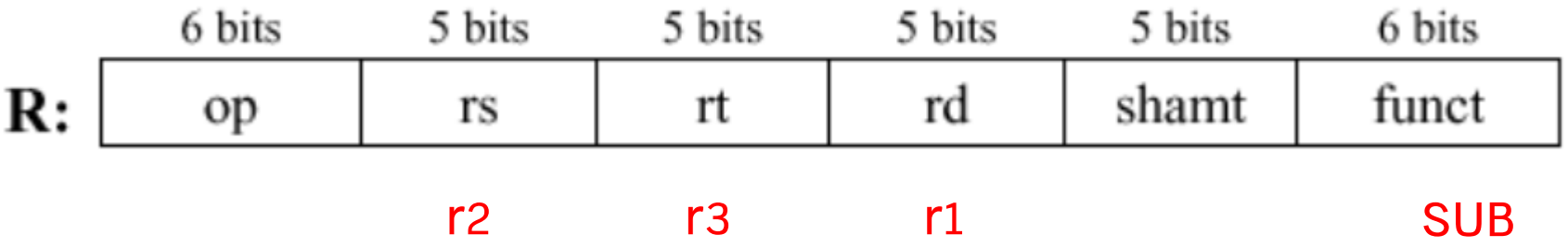
- **รีจิสเตอร์**
  - Read register 1: ฟิลด์ rs (5 บิต) จะถูกใช้เป็นหมายเลขของรีจิสเตอร์ตัวแรกที่จะถูกอ่าน
  - Read register 2: ฟิลด์ rt (5 บิต) จะถูกใช้เป็นหมายเลขของรีจิสเตอร์ตัวที่สองที่จะถูกอ่าน
  - Read data 1: ข้อมูลที่อ่านได้จากรีจิสเตอร์ rs จะถูกส่งออกไป
  - Read data 2: ข้อมูลที่อ่านได้จากรีจิสเตอร์ rt จะถูกส่งออกไป
  - Write register: ฟิลด์ rd (5 บิต) จะถูกใช้เป็นหมายเลขของรีจิสเตอร์ปลายทางที่จะเขียนผลลัพธ์กลับเข้าไป
  - RegWrite (สัญญาณควบคุมการเขียนรีจิสเตอร์): สัญญาณนี้ (มาจาก Control Unit ที่ไม่ได้แสดงในภาพนี้) จะถูกตั้งค่าเป็น 1 เพื่ออนุญาตให้มีการเขียนข้อมูลกลับไปยังรีจิสเตอร์

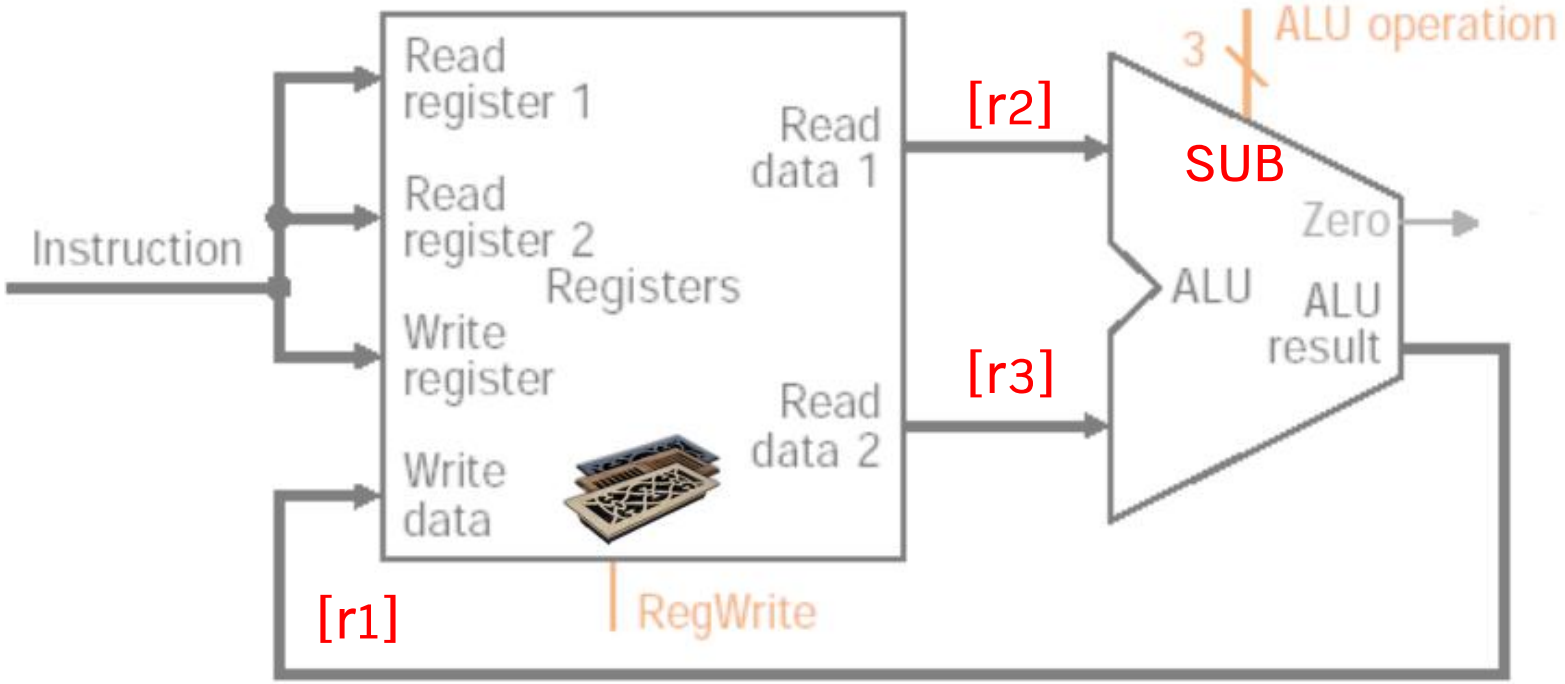
การทำงานของคำสั่ง R-Type



- ALU (Arithmetic Logic Unit)
- Read data 1 (จาก rs) และ Read data 2 (จาก rt) จะถูกป้อนเข้าสู่อินพุตของ ALU
- ALU operation (3 bits): สัญญาณควบคุมนี้ (ได้มาจากฟิลด์ funct ของคำสั่ง R-Type ผ่าน Control Unit) จะบอกให้ ALU ทำการดำเนินการเฉพาะอย่าง เช่น
  - 000 สำหรับ ADD (บวก)
  - 001 สำหรับ SUB (ลบ)
  - 010 สำหรับ AND
  - 011 สำหรับ OR
  - และอื่นๆ
- ALU result: ผลลัพธ์จากการดำเนินการของ ALU (เช่น ผลลัพธ์ของการบวกหรือลบ) จะถูกส่งออกมา
- Zero: เป็นสัญญาณบ่งชี้ว่าผลลัพธ์ของ ALU เป็นศูนย์หรือไม่

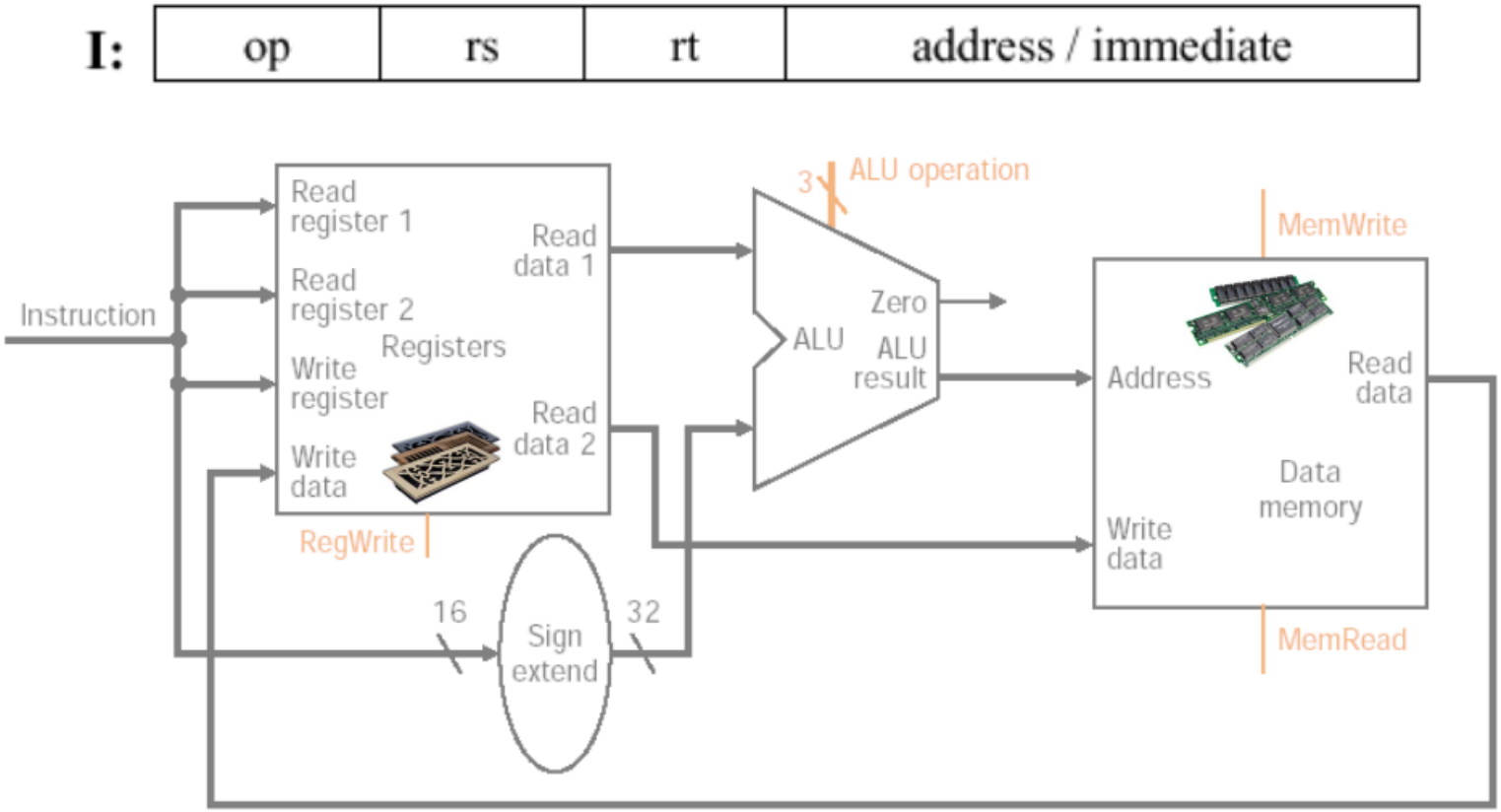
พิจารณาตัวอย่าง: r1 = r2 - r3





การบ้าน

อธิบายการทำงานของคำสั่ง Load-Store



Consider: r1 = M[r2 - 3]