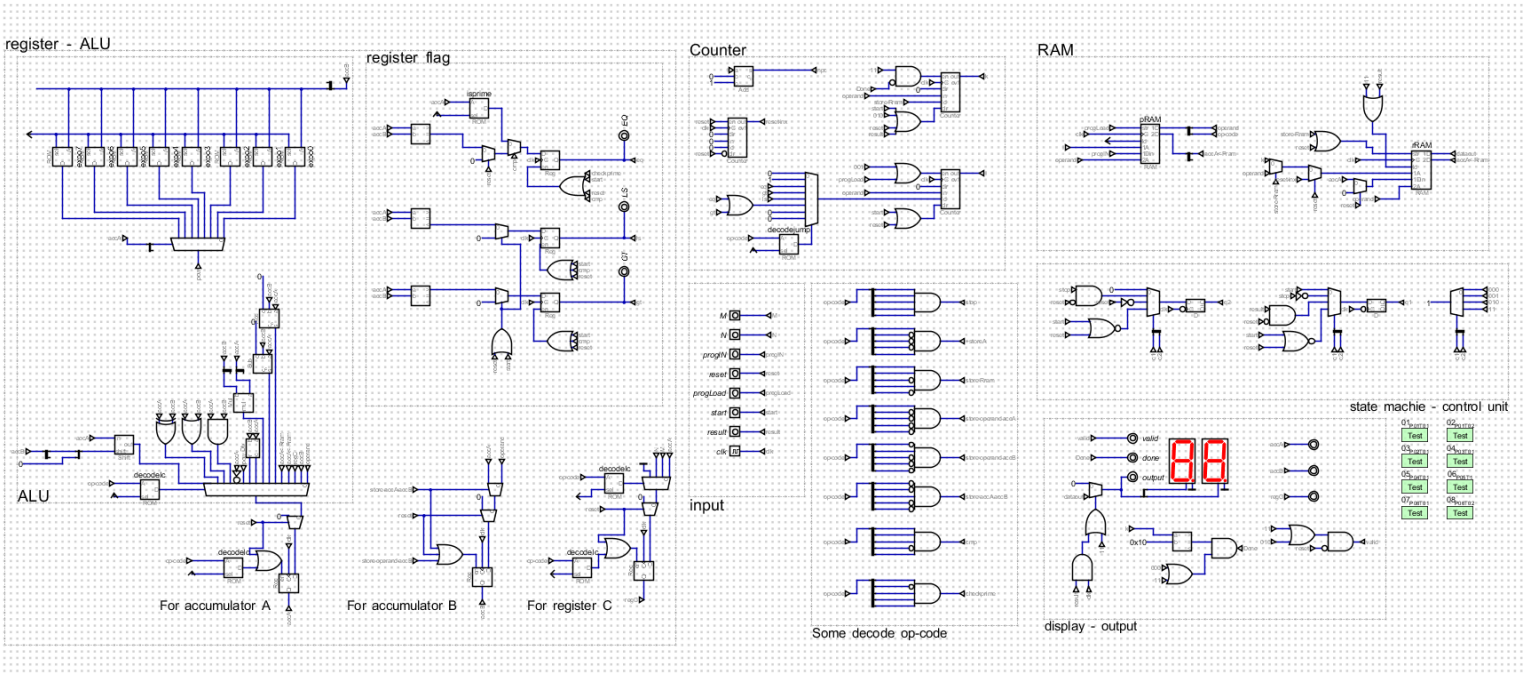
เอกสารด้านเทคนิค CEDT Final Project 2567

Digital Logic 2110252: Mini CPU

จัดทำโดยกลุ่ม DigLoEiEi



การออกแบบ CPU สำหรับโปรเจค Mini CPU

CPU ที่ออกแบบในโปรเจคนี้มีโครงสร้างการทำงานแบบ **Single-Cycle** ซึ่งทำให้สามารถประมวลผลคำสั่งได้อย่างมีประสิทธิภาพในหนึ่งรอบการทำงานของสัญญาณนาฬิกา (clock cycle) การทำงานของ CPU ถูกควบคุมด้วย **state machine** ที่ประกอบไปด้วยสี่สถานะหลัก ได้แก่ **Idle**, **Execute**, **Wait Result**, และ **Show Output** โดย CPU นี้ถูกออกแบบมาให้ทำงานร่วมกับหน่วยความจำ **Program RAM (pRAM)** และ **Result RAM (rRAM)** เพื่อจัดเก็บโปรแกรมคำสั่งและผลลัพธ์ตามลำดับ

โครงสร้างการทำงานแบบ State Machine

1. **Idle State**:
    - ทำหน้าที่รอรับคำสั่งจากโปรแกรม เมื่อได้รับสัญญาณ **start = 1** CPU จะเข้าสู่สถานะ **Execute** เพื่อเริ่มประมวลผลคำสั่งจากหน่วยความจำ **pRAM** ที่ตำแหน่ง **address 0x00** โดยที่ CPU ยังรอการส่งข้อมูลจาก **input** อื่น ๆ ที่อาจมีส่วนช่วยในการประมวลผลคำสั่ง
2. **Execute State**:
    - ในสถานะนี้ CPU จะอ่านคำสั่งจาก **pRAM** และดำเนินการตาม **op-code** ที่ระบุ โดยการทำงานจะเกี่ยวข้องกับการดำเนินการต่างๆ เช่น การคำนวณ การเคลื่อนย้ายข้อมูลระหว่าง **Accumulator A**, **Accumulator B**, และ **Register C** รวมถึงการควบคุมลำดับการทำงานด้วยการใช้คำสั่ง Jump การทำงานในสถานะนี้จะดำเนินไปจนกระทั่งพบคำสั่ง **STOP** (op-code 11111) จากนั้น CPU จะเข้าสู่สถานะ **Wait Result**
3. **Wait Result State**:
    - ในสถานะนี้ CPU จะหยุดการทำงานชั่วคราวและรอสัญญาณ **result = 1** เพื่อเข้าสู่ขั้นตอนการแสดงผลลัพธ์ เมื่อได้รับสัญญาณดังกล่าว CPU จะเข้าสู่สถานะ **Show Output**
4. **Show Output State**:
    - CPU จะนำค่าผลลัพธ์ที่เก็บไว้ใน **rRAM** แสดงออกทาง **output** รวมถึงบน **7-segment display** ที่มีอยู่สองตัว โดยผลลัพธ์จะถูกดึงออกมาจาก **rRAM** ตั้งแต่ตำแหน่ง address 0x00 - 0x0F หลังจากแสดงผลลัพธ์ครบแล้ว CPU จะกลับไปยังสถานะ **Idle** เพื่อรอคำสั่งใหม่

องค์ประกอบหลักของ CPU

- **Accumulator A (accA) และ Accumulator B (accB)**:
    - ทำหน้าที่เป็นหน่วยเก็บข้อมูลที่ใช้ในการประมวลผลคำสั่งทางคณิตศาสตร์ โดยมี op-code ที่ควบคุมการทำงาน เช่น **accA ⟵ Operand (00001)** สำหรับการนำค่า operand ไปเก็บใน **Accumulator A** และ **accB ⟵ Operand (00010)** สำหรับการนำค่าไปเก็บใน **Accumulator B**
- **Register C (regC)**:
    - เป็นหน่วยเก็บข้อมูลชั่วคราวที่สามารถรับข้อมูลจากทั้ง **accA** และ **accB** รวมถึงจาก input ภายนอกอย่าง **M** และ **N** ตัวอย่าง op-code ที่ควบคุมการทำงานของ **Register C** ได้แก่ **regC ⟵ accA (00101)** สำหรับการนำค่าจาก **accA** ไปเก็บไว้ใน **regC** และ **regC ⟵ M (00111)** สำหรับการนำค่าจาก input **M** ไปเก็บใน **Register C**
- **Arithmetic Logic Unit (ALU)**:
    - หน่วยประมวลผลทางคณิตศาสตร์ที่ทำงานร่วมกับ **accA** และ **accB** ในการดำเนินการต่างๆ เช่น การบวก การลบ การคูณ รวมถึงการดำเนินการทางตรรกศาสตร์ (logical operations) ตัวอย่างเช่น op-code **accA ⟵ accA + accB (10001)** ใช้ในการนำค่าจาก **accA** มาบวกกับ **accB** แล้วเก็บผลลัพธ์ไว้ที่ **accA**
- **Program RAM (pRAM)** และ **Result RAM (rRAM)**:
    - **pRAM** ทำหน้าที่เก็บโปรแกรมคำสั่งที่ส่งเข้ามา โดยเริ่มต้นจาก address **0x00** และทำการประมวลผลตามลำดับของคำสั่งจนกว่าจะพบคำสั่ง **STOP** ที่จะทำให้การประมวลผลหยุดลง
    - **rRAM** ใช้เก็บผลลัพธ์จากการคำนวณ โดยผลลัพธ์ที่ได้จะถูกแสดงในช่วง address **0x00 - 0x0F** ซึ่ง CPU จะดึงข้อมูลจาก **rRAM** เพื่อแสดงผลในขั้นตอนสุดท้าย
- **ROM และ MUX**:
    - **ROM** เก็บข้อมูลเกี่ยวกับ **op-code** ที่จำเป็นสำหรับการควบคุมการทำงานของ CPU ในแต่ละคำสั่ง

- **MUX (Multiplexer)** ทำหน้าที่เลือกเส้นทางของข้อมูลที่เหมาะสมตาม **op-code** ที่ระบุ เช่น การเลือกแหล่งข้อมูลจาก **accA**, **accB** หรือ **Register C** เพื่อส่งไปยัง ALU หรือหน่วยความจำอื่นๆ

Inputs ของ CPU

CPU นี้มี **inputs** ดังต่อไปนี้:

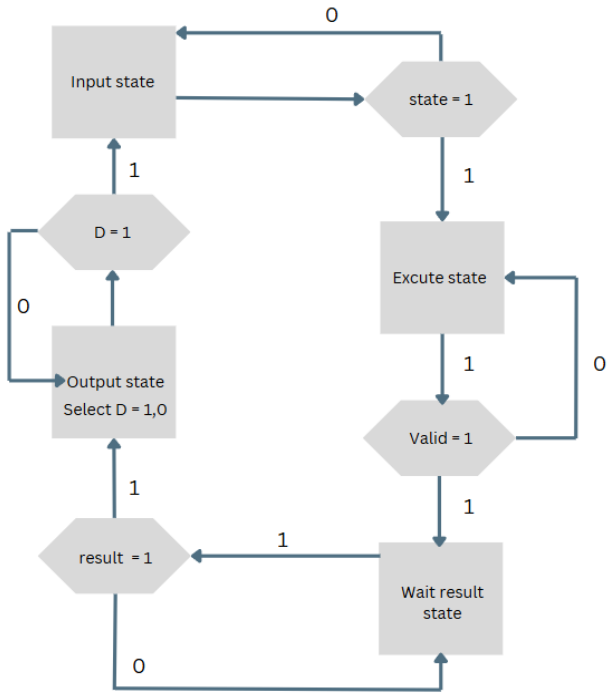
- **M (8 bits)**: พารามิเตอร์ที่ 1 ซึ่งจะใช้ในการประมวลผลคำสั่ง
- **N (8 bits)**: พารามิเตอร์ที่ 2 ซึ่งจะใช้ในการประมวลผลคำสั่ง
- **progIN (13 bits)**: ข้อมูล input ที่ใช้ส่งโปรแกรมเข้ามาทีละคำสั่งจนกระทั่งครบก่อนจะเริ่มทำงาน
- **reset (1 bit)**: เมื่อได้รับค่า 1 จะทำการ reset การทำงานของวงจรและเคลียร์ค่าของ **pRAM** และ **rRAM** ให้เป็น 0 ทั้งหมด
- **progLoad (1 bit)**: เมื่อได้รับค่า 1 จะเริ่มอ่านค่าจาก **progIN** ทีละคำสั่ง ไปเก็บใน **pRAM** เริ่มตั้งแต่ address **0x00** จนกว่าจะมีสัญญาณ **progLoad** เป็น 0 โดยสามารถเก็บคำสั่งได้สูงสุดถึง 256 คำสั่ง
- **start (1 bit)**: เมื่อได้รับค่า 1 จะเริ่มทำงานตามคำสั่งที่อยู่ใน **pRAM**
- **result (1 bit)**: เมื่อได้รับค่า 1 จะเริ่มส่งคำตอบที่อยู่ใน **rRAM**
- **clk (1 bit)**: สัญญาณ clock ใช้สำหรับให้จังหวะในการทำงานของวงจร

วงจรมี output ที่สำคัญดังนี้:

- **valid (1 bit)**: จะมีค่าเป็น 1 เมื่อทำงานตามโปรแกรมเสร็จเรียบร้อยและพร้อมส่งคำตอบ
- **done (1 bit)**: จะมีค่าเป็น 1 เมื่อการทำงานเสร็จสิ้นและส่งผลคำตอบเรียบร้อย หรือตอนเริ่มต้นครั้งแรกที่พร้อมรับคำสั่ง reset, progLoad, หรือ start
- **output (8 bits)**: แสดงค่าผลลัพธ์ที่ได้จากการทำงาน โดยจะแสดงผลลัพธ์ที่อยู่ใน **rRAM** เป็นเลขฐาน 16 หลังจากได้รับสัญญาณ result
- **7-segment display (2 ตัว)**: ใช้สำหรับแสดงค่าผลลัพธ์ของ output เพื่อให้ตรวจสอบผลลัพธ์ได้
- **rRAM (ขนาด 256 x 8 bits)**: ใช้สำหรับเก็บคำตอบ

ASM Chart

ASM chart โดยย่อสำหรับการออกแบบ Mini CPU ที่ใช้ State Machine



ภาพการออกแบบASM chartของโปรแกรม

การอธิบาย ASM Chart

- **Idle State**: ในสถานะนี้ CPU รอรับข้อมูลจากผู้ใช้ โดยไม่มีสัญญาณ reset เนื่องจากไม่จำเป็นต้องรีเซ็ตเมื่ออยู่ในสถานะ Idle หากได้รับสัญญาณ start CPU จะเปลี่ยนไปที่ **Execute State** เพื่อเริ่มทำงานตามคำสั่งที่ได้รับจาก **pRAM**.
- **Execute State**: CPU จะทำการประมวลผลคำสั่งที่ได้รับจาก **pRAM** โดยจะตรวจสอบว่ามีคำสั่ง STOP หรือไม่ หากมีคำสั่งนี้ CPU จะเปลี่ยนไปที่ **Wait Result State** เพื่อรอผลลัพธ์ ในกรณีที่มีสัญญาณ reset CPU จะกลับไปที่ **Idle State** เพื่อเริ่มต้นใหม่.
- **Wait Result State**: CPU จะรอจนกว่าจะได้รับสัญญาณ result มีค่าเป็น 1 เพื่อส่งค่าผลลัพธ์ที่อยู่ใน **rRAM** ไปยังผู้ใช้ ถ้ามีสัญญาณ reset CPU จะเปลี่ยนกลับไปที่ **Idle State**.

- **Show Output State**: ในสถานะนี้ CPU จะส่งค่าผลลัพธ์จาก **rRAM** ไปยังผู้ใช้ เมื่อเสร็จสิ้นการส่งข้อมูล CPU จะกลับไปที่ **Idle State** หรือถ้ามีสัญญาณ reset จะทำการกลับไปที่ **Idle State** เช่นกัน.

Control Unit (CU) ของ Mini CPU

Control Unit (CU) ของ Mini CPU ได้รับการออกแบบเพื่อให้เป็นศูนย์กลางในการควบคุมและประสานการทำงานของหน่วยประมวลผล โดยทำหน้าที่ควบคุมลำดับการทำงานของคำสั่งในโปรแกรมตามโครงสร้างที่กำหนดใน ASM Chart

1. โครงสร้างของ Control Unit

- **State Machine:** CU ถูกสร้างขึ้นในรูปแบบของ state machine ซึ่งประกอบด้วย 4 สถานะหลัก ได้แก่ Idle, Execute, Wait Result, และ Show Output เพื่อจัดการกับการทำงานในแต่ละขั้นตอน
- **Logic Circuit:** วงจรตรรกะภายใน CU ใช้ในการประมวลผลสัญญาณจาก input ต่าง ๆ เช่น reset, start, result และ progLoad โดยจะควบคุมการทำงานของ CPU ตามคำสั่งที่ได้รับ
- **MUX (Multiplexer):** MUX ถูกใช้เพื่อเลือก op-code ที่จะส่งไปยัง ALU หรือ registers ตามความต้องการของคำสั่ง เพื่อให้การทำงานมีความยืดหยุ่นและเหมาะสม
- **Clock Signal:** สัญญาณนาฬิกา (clk) มีบทบาทในการซิงโครไนซ์การทำงานของส่วนต่าง ๆ ใน CPU ซึ่งช่วยให้การประมวลผลเป็นไปอย่างมีระเบียบและแม่นยำ

2. การทำงานของ Control Unit

- **รับข้อมูลจาก input:** CU จะตรวจสอบสัญญาณ input อย่างต่อเนื่อง เช่น reset ที่จะทำให้กลับไปสู่สถานะ Idle, start ที่จะเริ่มการทำงานตามคำสั่งใน pRAM, และ result ที่จะส่งผลลัพธ์ออกมา
- **การส่งคำสั่งไปยัง ALU และ Registers:** เมื่อตรวจสอบว่าเริ่มทำงานแล้ว CU จะส่งสัญญาณที่เหมาะสมไปยัง ALU เพื่อให้ทำการคำนวณ หรือส่งข้อมูลไปยัง accumulators (accA และ accB) และ register C ตามคำสั่งใน op-code

- **การควบคุมการทำงานของ rRAM:** CU ควบคุมการเก็บและส่งค่าผลลัพธ์ไปยัง rRAM โดยกำหนดวิธีการเก็บข้อมูลจาก accA และ accB ลงใน rRAM ตามคำสั่งที่ระบุในโปรแกรม

Data Path ของ Mini CPU

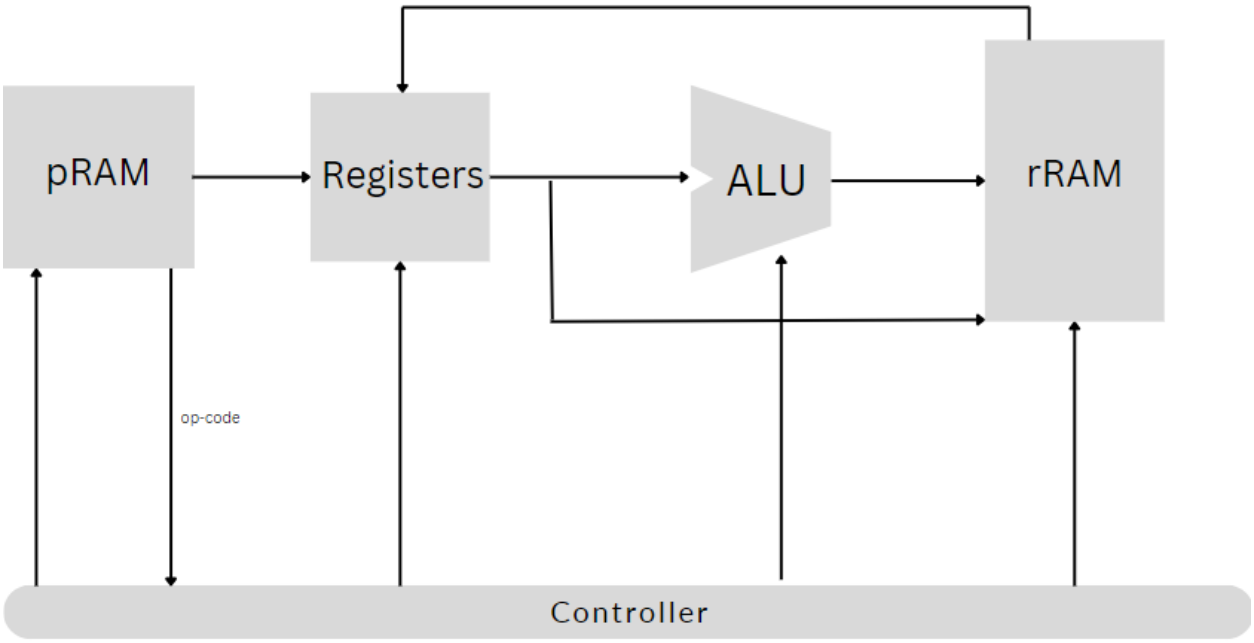
Data Path ของ Mini CPU เป็นโครงสร้างที่ใช้ในการจัดการและส่งผ่านข้อมูลระหว่างส่วนต่าง ๆ ของ CPU เพื่อให้การทำงานของหน่วยประมวลผลมีประสิทธิภาพและรวดเร็ว การออกแบบ Data Path จะต้องเชื่อมโยงระหว่าง ALU, registers, memory (pRAM และ rRAM) และ counters อย่างเหมาะสม

Data Path ประกอบด้วย:

1. **ALU (Arithmetic Logic Unit)**: ALU เป็นส่วนที่ทำการคำนวณทางคณิตศาสตร์และตรรกะ โดยจะรับข้อมูลจาก accumulators (accA และ accB) และส่งผลลัพธ์กลับไปยัง accA
2. **Registers**: Data Path ประกอบด้วย accumulators (accA และ accB) และ register C ที่ใช้ในการเก็บค่าต่าง ๆ ขณะทำงาน ค่าที่เก็บใน accumulators จะถูกใช้ในการคำนวณใน ALU และ register C จะเก็บค่าที่สำคัญจากการประมวลผล
3. **Memory (pRAM และ rRAM)**:
    - **pRAM (Program RAM)** ใช้สำหรับเก็บโปรแกรมที่ต้องการประมวลผล โดยจะถูกเข้าถึงผ่าน counter i ที่ใช้ควบคุม address ในการอ่านและเขียนข้อมูล
    - **rRAM (Result RAM)** ใช้สำหรับเก็บผลลัพธ์ที่ได้จากการคำนวณ ซึ่งจะเข้าถึงผ่าน counter k ที่ทำหน้าที่ควบคุม address ของ rRAM
4. **การทำงานของ Data Path**:
    - **การรับข้อมูล**: Data Path เริ่มต้นด้วยการรับข้อมูลจาก input ที่เข้ามา เช่น M, N, progIN เป็นต้น ซึ่งข้อมูลเหล่านี้จะถูกส่งไปยัง registers หรือ pRAM ตามที่กำหนด
    - **การส่งข้อมูลไปยัง ALU**: ข้อมูลที่อยู่ใน accA และ accB จะถูกส่งไปยัง ALU เพื่อทำการคำนวณตามคำสั่งที่ได้รับจาก Control Unit ข้อมูลที่ได้จะถูกส่งกลับไปยัง accA

- **การเขียนและอ่านจาก Memory**: Data Path จะจัดการกับการอ่านข้อมูลจาก pRAM โดยใช้ counter i ในการระบุ address ที่จะอ่านคำสั่งตามลำดับ ข้อมูลที่อ่านจะถูกนำไปเก็บใน accumulators หรือ registers ตามคำสั่งที่กำหนด และสำหรับ rRAM การใช้ counter k จะควบคุม address ที่ใช้เก็บผลลัพธ์จากการประมวลผลที่เกิดขึ้น
- **การส่งผลลัพธ์ออก**: ผลลัพธ์ที่ได้จากการประมวลผลจะถูกเก็บใน rRAM และสามารถส่งออกไปยัง output ได้ โดยจะแสดงผลผ่าน 7-segment display หรือ output ขนาด 8 bits

5. **การเชื่อมต่อระหว่างส่วนต่าง ๆ**:
   - **MUX (Multiplexer)**: MUX ใช้ในการเลือกข้อมูลที่ต้องการส่งไปยัง ALU หรือ registers โดยการเลือกผ่าน op-code ที่จะทำงาน



รูปภาพ Data Path โดยย่อ