CEDT Final Project 2567 Digital Logic 2110252: Mini CPU

สมาชิกกลุ่ม
พัลลภ บุญขัน 6733174921
จิรัชญา กัญญะพิลา 6733032421
สินสุดา รักษ์พอเพียง 6733271021
ธมกร ขัมพานนท์ 6733104021

แนวคิดการออกแบบ

Mini CPU ของกลุ่ม iluvdcl มีเป้าหมายหลักอยู่ 2 ประเด็น ได้แก่ 1.ต้องการให้สามารถทำงานตาม instruction ทั้งหมดได้โดยใช้เวลาน้อยที่สุด และ ไม่เกิด data hazard โดย cpu ของเราจะใช้ 2 clock cycle และเป็นแบบ rising edge ต่อ 1 instruction คือ fetch และ write แต่หากทำอย่างที่กล่าวไว้ข้างต้นมีโอกาสเกิด data hazard ได้ ผู้จัดทำจึงเปลี่ยนให้ write ก่อนแล้วจึง fetch คำสั่งต่อไป ประเด็นที่ 2 คือ Mini CPU สามารถมองจากภายนอกแล้วเข้าใจได้โดยทันที ว่าแต่ละส่วนทำหน้าที่อะไร และหากส่วนไหนมีปัญหา สามารถแก้ส่วนนั้น ได้โดยไม่ต้องคำนึงว่า จะเกี่ยวโยงกับส่วนอื่นมาก โดยจะแยกเป็น 4 ส่วนหลัก ๆ ดังนี้

- Control Unit
  โดยมีองค์ประกอบคล้ายกับ cpu ทั่วๆไปที่มี PC กับ IR :
  PC บอก Address ของคำสั่งใน pRAM แล้วเพิ่มไปเรื่อยๆ และ IR จะนำคำสั่งที่มาจาก pRAM ไป execute และจะทำการ fetch คำสั่งทุก 1 clock

- ALU
  ในส่วนนี้จะแยกดึงคำตอบของแต่ละ instruction ตาม opcode โดยจำนวนวงจรมีปริมาณตามจำนวน opcode เพื่อความเร็วจึง ทำให้ทุกวงจร ทำการคำนวณแบบ parallel แล้ว output มาเข้า mux ที่จะเลือกคำตอบ โดยอิงจาก opcode ที่กำลังทำงานอยู่ แล้วไปต่อเข้า กับ register และ memory

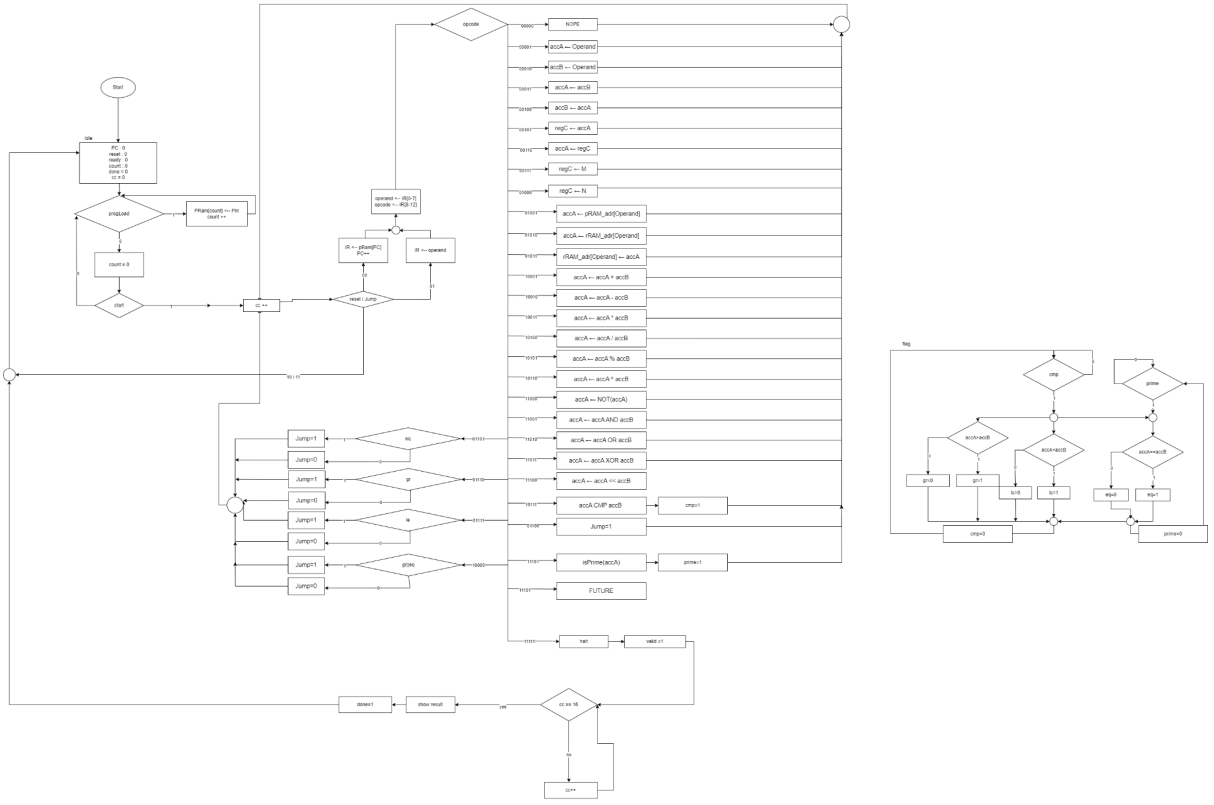
- Register
  ในส่วนนี้จะ ให้ databus สายเดียวเชื่อมกับทุก register แต่ไม่สามารถเขียนลง register ในทันที ตัวที่จะบอกว่า register ไหนมีสิทธิ์เขียนจะมาจาก opcode แต่ในกรณีดึงค่าจาก register สามารถดึงได้เลย ทำให้ต้องมีการ เช็คอีกทีว่า จะใช้จริงๆรึเปล่าจาก opcode

- Memory
  สำหรับส่วนนี้ จะทำงานเหมือน Register แต่ที่เพิ่มมาคือ วงจรที่ทำให้สามารถ ส่งคำตอบเมื่อ result เป็น 1 และจะวนลูป 16 ครั้ง เพื่อส่งคำตอบตั้งแต่ Address 0x00 - 0x0F และสามารถ วนลูป reset ค่าจาก Address 0x00 - 0x0F ได้

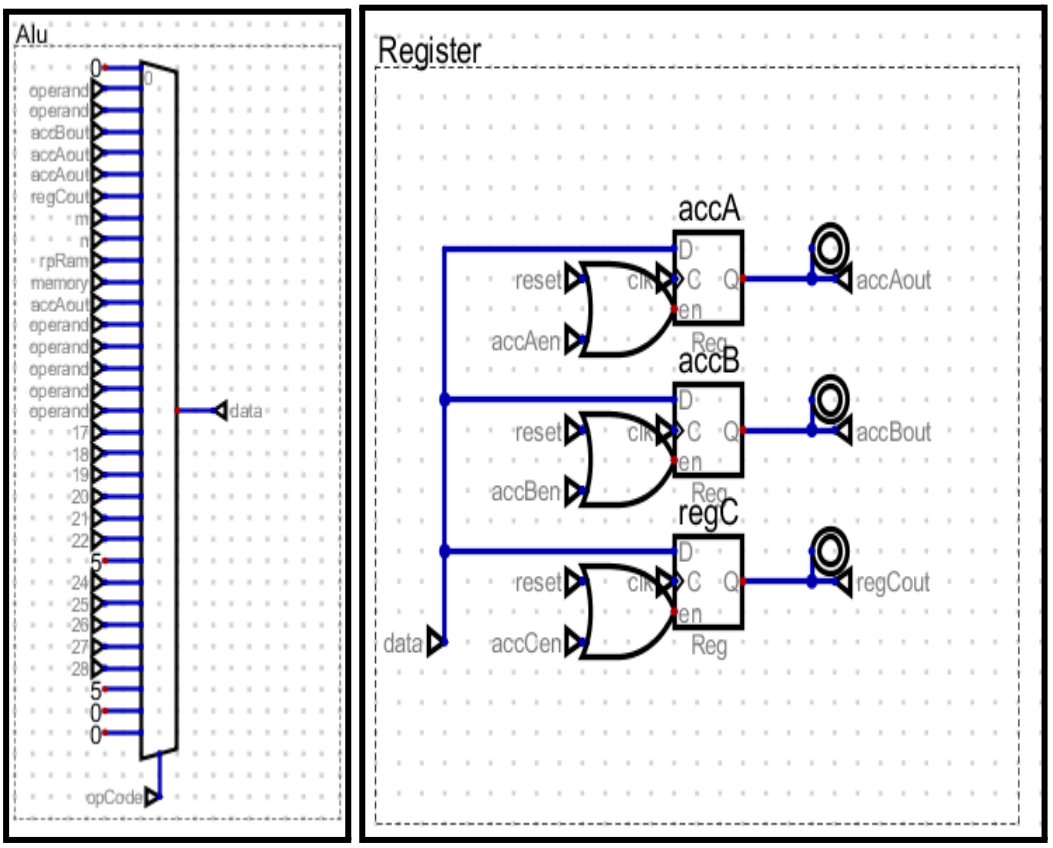
- LCD
  ส่วนนี้เป็นคำสั่งใหม่ที่เพิ่งได้โจทย์มา เราทำเป็น service แยก มีที่เก็บข้อมูลของตัวเอง และประมวลผล เขียน อ่าน ในตัวเอง
  - โดยโหลด ค่า m n ไว้ก่อน มาเก็บไว้ใน register เฉพาะ 2 ตัว

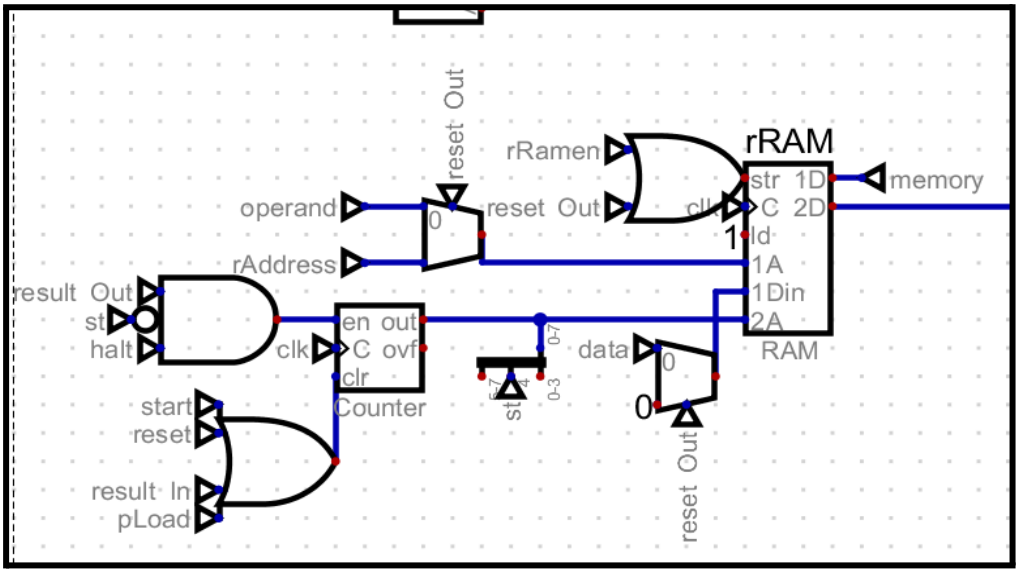
- เมื่อโหลดเสร็จแล้วจึงเริ่มหาค่า gcd
- เมื่อได้ค่า gcd ก็จะหาค่า lcm ได้จากสูตร (m * n) / gcd
- เก็บค่า lcd ลงใน rRAM โดยใช้ counter ทำงาน สองครั้ง
  - ครั้งแรก counter output 0 ละเอามาบวกกับ const 0E
  - ครั้งที่สอง counter output เพิ่มมา 1 เอามาบวก const 0E
  ได้เป็นตำแหน่งที่จะเก็บค่าใน rRAM

ASM Chart



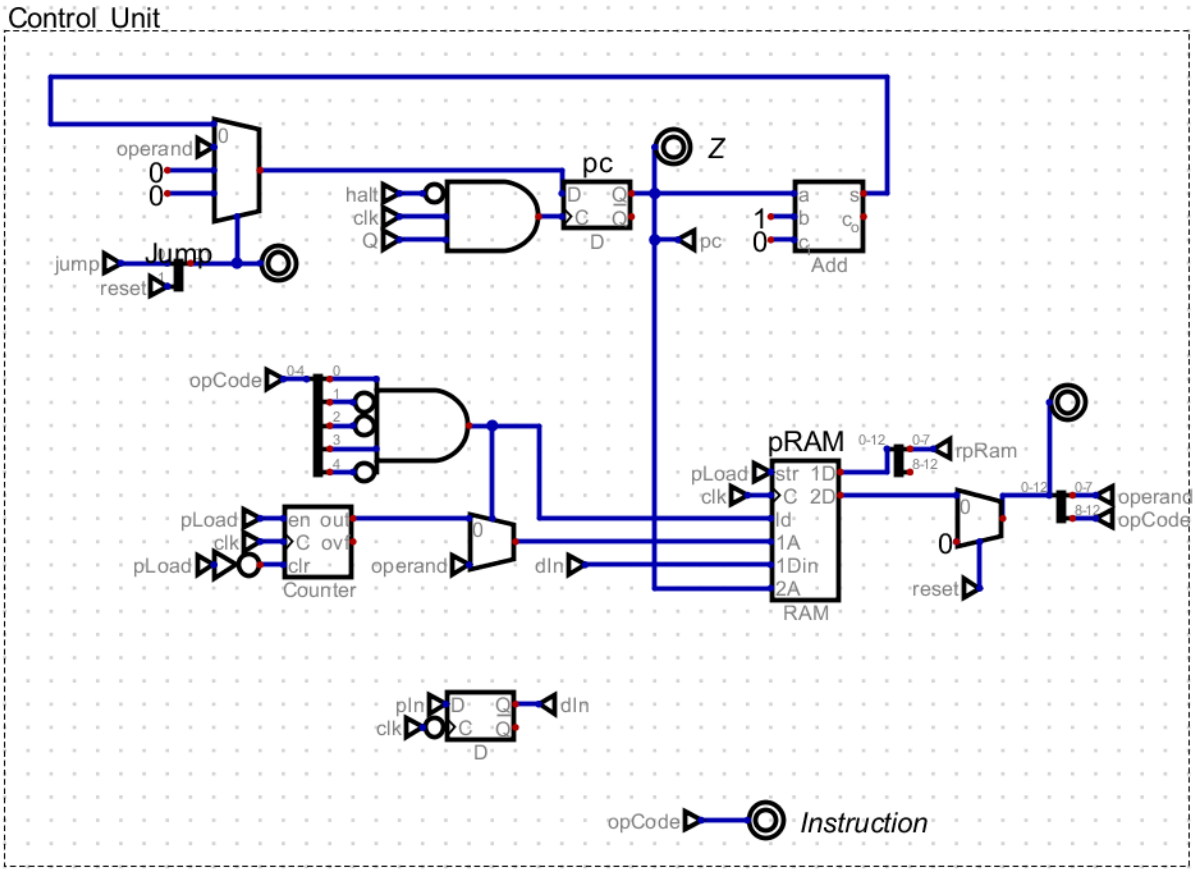
Data Path


Alu
operand
operand
accBout
accAout
accAout
regCout
m
n
rpRam
memory
accAout
operand
operand
operand
operand
operand
17
18
19
20
21
22
5
24
25
26
27
28
5
0
0
opCode
data
Register
accA
reset
clk
accAen
D
C
Q
en
Reg
accAout
accB
reset
clk
accBen
D
C
Q
en
Reg
accBout
regC
reset
clk
data
accCen
D
C
Q
en
Reg
regCout


reset Out
rRamen
rRAM
operand
reset Out
str 1D
C 2D
ld
1A
1Din
2A
RAM
memory
rAddress
result Out
st
halt
en out
clk
C ovf
clr
Counter
start
reset
result In
pLoad
data
reset Out

- ข้อมูล 8 bits ที่ได้จากการ execute ของแต่ละ instruction จะออกมาจาก ALU เกือบทั้งหมด ยกเว้น opcode ที่ต้อง jump หรือทำงานเกี่ยวกับ flag โดย
- ข้อมูลที่ได้จากการคำนวณจะเชื่อมกับ Register และ rRAM โดยตรง แต่ต้องมีสิทธิ์เขียน หรือ port str มีค่า 1 เข้าจึงจะสามารถเปลี่ยนแปลงค่าของ storage นั้นได้

Control Unit



ในส่วนของ control unit จะมีองค์ประกอบหลักๆอยู่ สามส่วนคือ PC, IR และ pRAM เป้าหมายของส่วนนี้หลักๆคือไม่ต้องการ pc มีช่วงมีช่วงที่หยุดทำงานน้อยที่สุด และไม่ทำให้เกิด data hazard ซึ่งถ้ามีมากจะทำให้ cpu ทำงานได้ช้า ดังนั้นจึงต้องทำให้ pc ใช้ clock น้อยที่สุดก่อนที่จะ fetch คำสั่งถัดไป ต่อมาจะเป็นรายละเอียดของ ส่วนประกอบของ control unit

- PC

  หน้าที่หลักๆคือ ทำการ fetch ข้อมูลที่มีทั้ง operand และ opcode ซึ่งอยู่ใน pRAM ออก port 2D ซึ่งจะกำหนดให้เป็นส่วนของ IR ที่จะบอกว่า cpu ต้องทำอะไรต่อจากนั้น

  โดยจะออกแบบให้ pc สามารถทำได้สามอย่างคือ 1) fetch คำสั่งถัดไป 2) jump ไปยัง address ต่างๆ ซึ่งเมื่อมีการ reset pc จะเข้าสู่ idle state หมายความว่าไม่ต้อง fetch คำสั่งถัดไปแล้ว

- IR

  ในส่วนนี้ จะทำเพียงแค่กระจาย สัญญาณ 13 bit ที่ PC fetch มาแยกไปเป็น operand และ opcode แต่ไม่ใช้ clock และในกรณีที่มีการกด reset จะทำการ เก็บค่าเป็น 0 (13 bits) แทน และไม่สนว่า PC จะ fetch ค่าอะไรมา หมายความว่าให้ cpu ไม่ต้องทำอะไร

- pRAM

  pRAM จะเป็นตัวเก็บ instruction ที่โหลดเข้ามาก่อนที่ cpu จะเริ่มทำงาน เพื่อให้ PC fetch ไปใช้งาน โดยกลุ่มเรา กำหนดให้ port 1A เป็น port สำหรับเก็บ instruction และ ใช้สำหรับทำงานกับบาง opcode ส่วน port 2A จะเป็น port สำหรับ fetch คำสั่งเข้า IR