สมาชิกกลุ่ม
1 6833122321 ธีร์จุฑา สิทธิสุขเอนก
2 6833076121 ณัฏฐณิชา ศรบัณฑิต
3 6833266021 สิรวิชญ์ กาญจนคีรีธำรง
4 6833225321 ภูริสรณ์ ตลึงจิตร

แนวคิด

1.เป็น mini CPU ที่ทำงานตาม instruction ได้โดยใช้ clock น้อยที่สุดและไม่เกิด data hazard
2.ต่อ 1 instruction มีการทำงาน fetch และ write ซึ่ง write ก่อนค่อย fetch คำสั่งถัดไป
3.สามารถมองและเข้าใจวงจรได้ง่าย
4.ทำแต่ละส่วนแยกกัน เพื่อสามารถแก้ไขส่วนที่มีปัญหาได้โดยตรง

Control Unit

โดยมีองค์ประกอบคือ PC IR และ pRAM
PC บอก Address ของค่าสั่งใน pRAM แล้วเพิ่มไปเรื่อยๆ และ IR จะนำค่าสั่งที่มาจาก pRAM ไป execute และจะทำการ fetch ค่าสั่งทุก 1 clock

ALU

ในส่วนนี้จะแยกแต่ละ instruction ตาม opcode โดยจำนวนวงจรมีเท่ากับจำนวน opcode เพื่อความเร็วจึงทำการคำานวณแบบ parallel แล้ว output มาเข้า mux ที่จะเลือกค่าตอบ โดยอิงจาก opcode ที่กำลังทำงาน อยู่ แล้วไปต่อเข้า กับ register และ memory

Register

ในส่วนนี้จะ ให้ databus ที่เก็บค่าที่เลือกมาจาก ALU เชื่อมกับทุก register แต่ไม่สามารถเขียนลง register ได้จนกว่า จะมีตัวที่จะบอกว่า register ไหนมีสิทธิ์เขียนซึ่งจะมาจาก opcode แต่ในกรณีที่บาง opcode จำเป็นต้องตั้งค่าเพื่อใส่ค่าอื่นจาก databus ทำให้ต้องมีการ เช็คอีกทีว่า เป็น opcode พิเศษตัวนั้นไหม

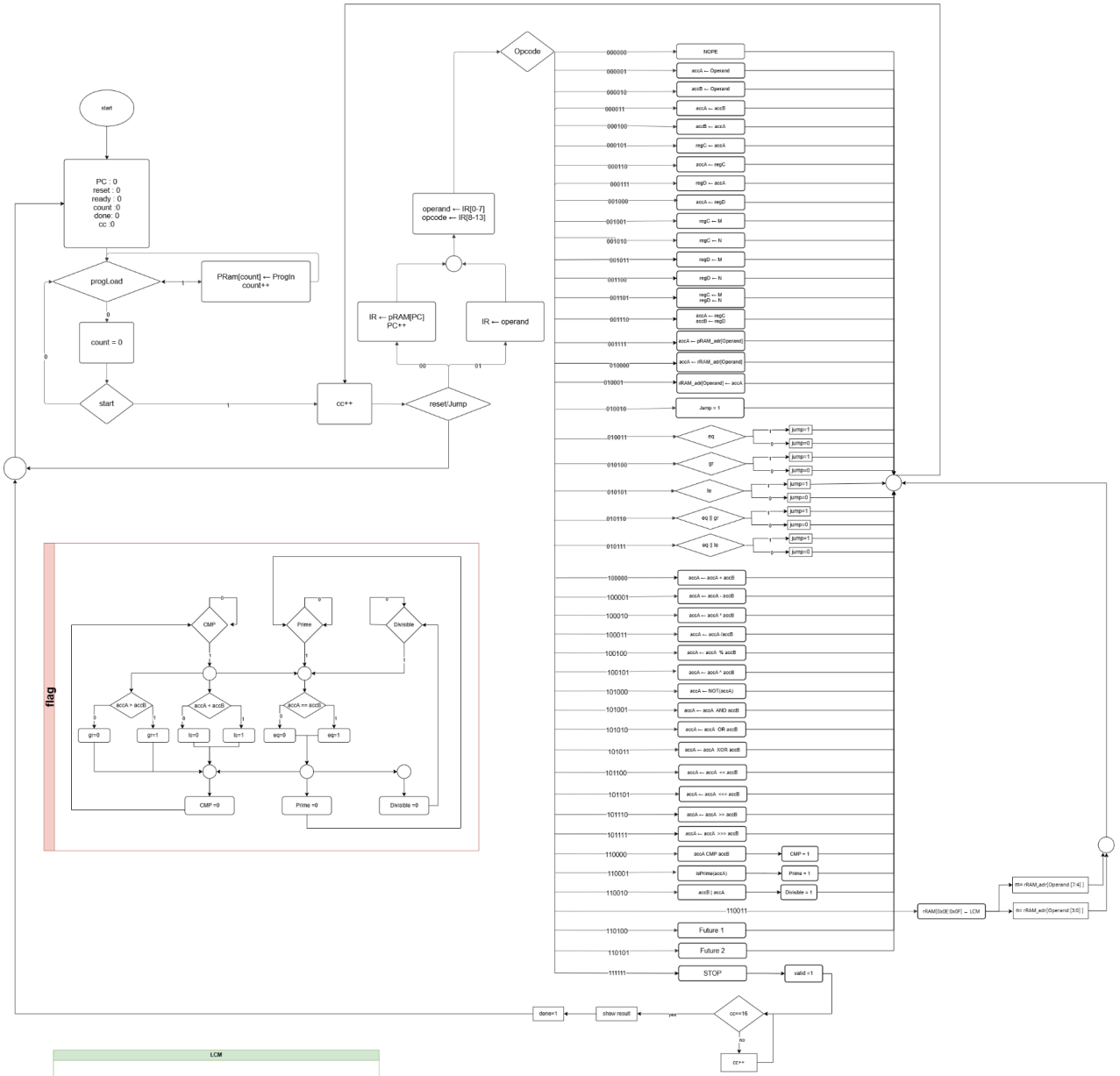
Memory

สำหรับส่วนนี้ จะทำงานเหมือน Register แต่ที่เพิ่มมาคือ วงจรที่ทำให้สามารถ ส่งคำ ตอบเมื่อ result เป็น 1 และจะวนลูป 16 ครั้ง เพื่อส่งคำตอบตั้งแต่ Address 0x00 - 0X0F และสามารถวนลูป reset ค่าจาก Address 0x00 - 0x0F ได้

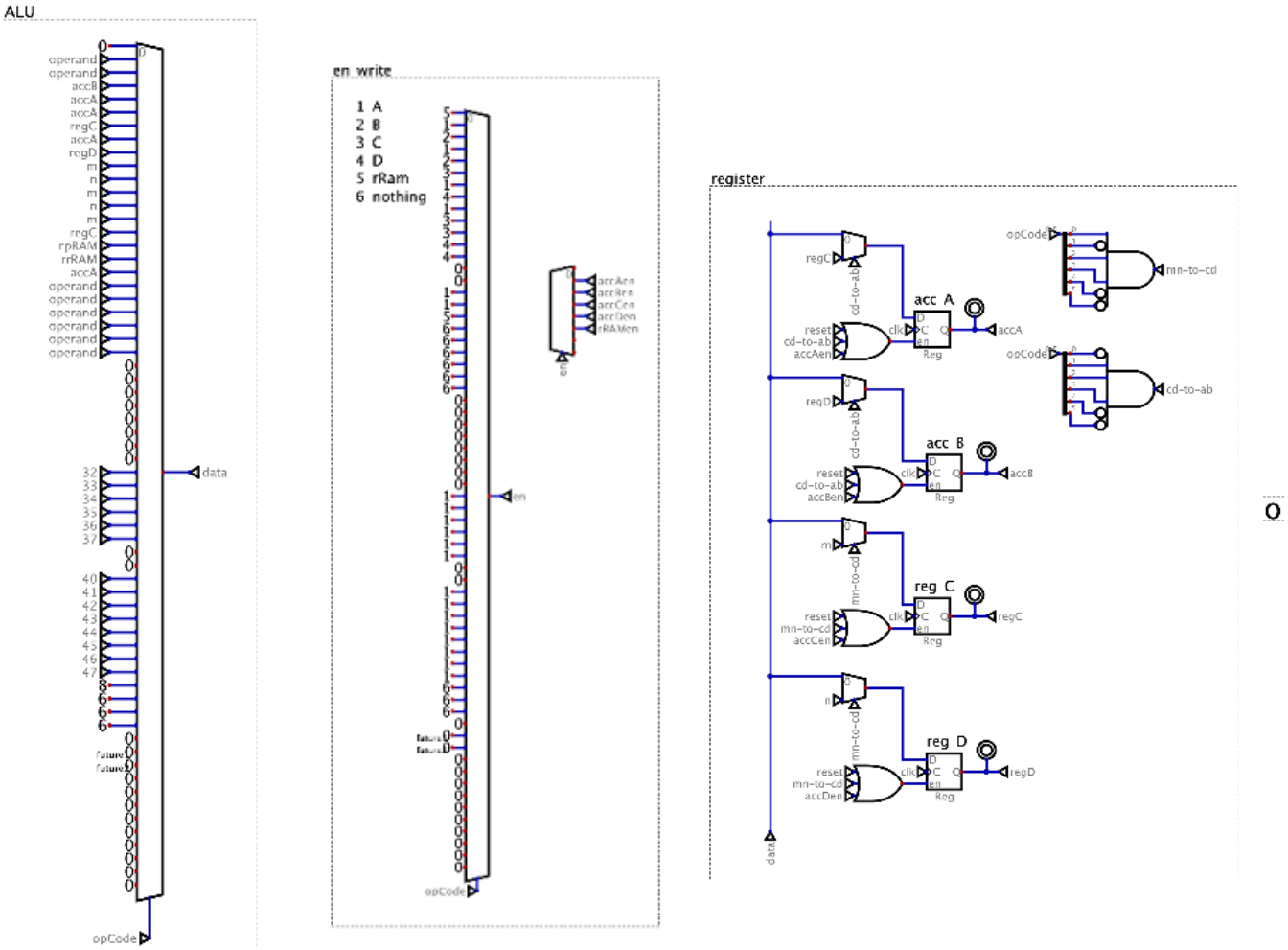
LCM (ไม่สำเร็จ)

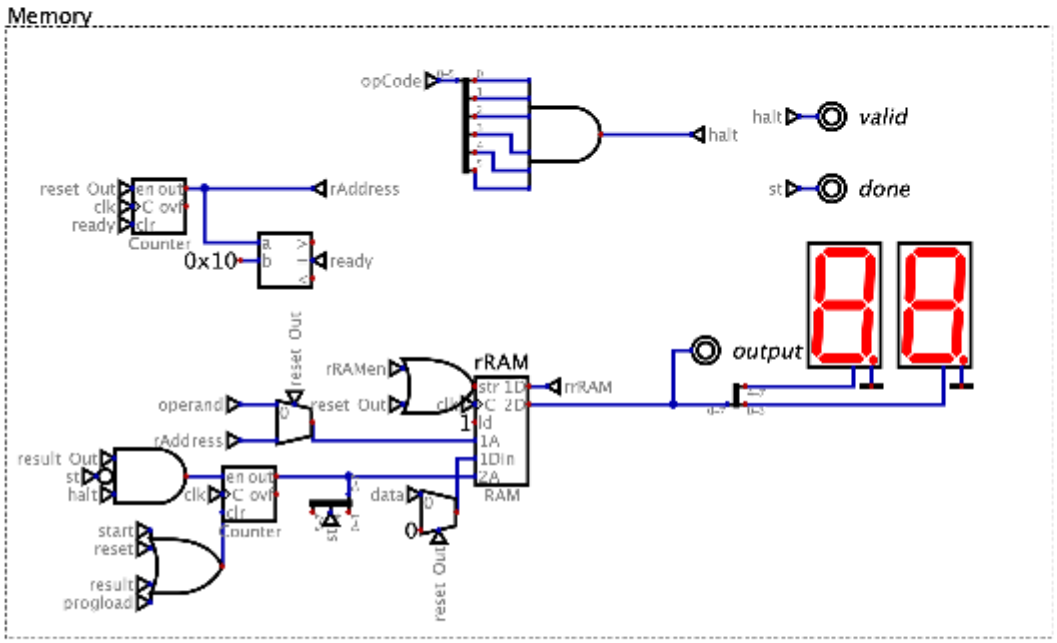
เป็นหนึ่งใน opcode ธรรมดา โดยแนวคิดที่ใช้คือการใช้ ROM เข้ามาช่วยในการทำ ค.ร.น. ให้โดยใส่เลข 2 ตัวขนาด 8 bit ที่ต้องการหาไป และ output ออกมาเป็น 16 bit โดย 8 bit หน้าคือค่าของ most significant และ 8 bit หลังคือค่าของ least significant ซึ่งเกิดจากการกำหนดค่าข้างใน ROM หลังจากนั้นทำการเก็บ 2 ค่านี้ที่ตำแหน่ง rRAM[0x0E] และ rRAM[0x0F] ตามลำดับ
-ปัญหา คือ ยังไม่สามารถรับค่าที่ต้องการหาค.ร.น. มาได้ทั้งหมด โดยหาได้เพียงแค่ตัวแรก อีกตัวที่ได้มายังคงเป็น address เก่าทำให้ค่าของตัวที่สองมีค่าเท่ากับตัวแรก ทำให้ค่าของค.ร.น.ยังไม่ถูกต้อง

ASM CHART

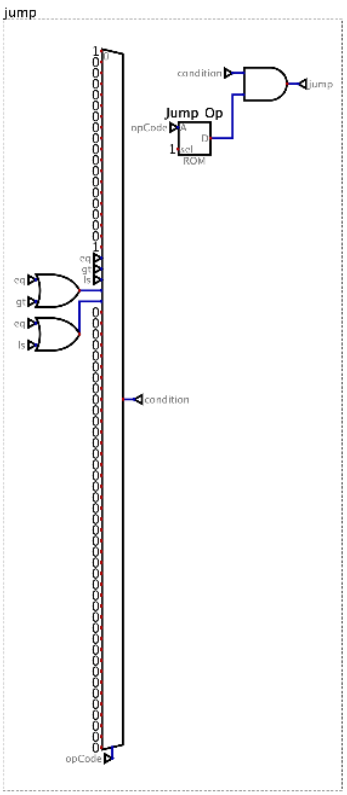

start
PC : 0
reset : 0
ready : 0
count :0
done: 0
cc :0
progLoad
PRam[count] ← ProgIn
count++
count = 0
start
cc++
reset/Jump
IR ← pRAM[PC]
PC++
IR ← operand
operand ← IR[0-7]
opcode ← IR[8-13]
Opcode
000000 NOPE
000001 accA ← Operand
000010 accB ← Operand
000011 accA ← accB
000100 accB ← accA
000101 regC ← accA
000110 accA ← regC
000111 regD ← accA
001000 accA ← regD
001001 regC ← M
001010 regC ← N
001011 regD ← M
001100 regD ← N
001101 regC ← M
regD ← N
001110 accA ← regC
accB ← regD
001111 accA ← pRAM_adr[Operand]
010000 accA ← rRAM_adr[Operand]
010001 rRAM_adr[Operand] ← accA
010010 Jump = 1
010011 eq
010100 gr
010101 le
010110 eq || gr
010111 eq || le
jump=1
jump=0
100000 accA ← accA + accB
100001 accA ← accA - accB
100010 accA ← accA * accB
100011 accA ← accA /accB
100100 accA ← accA % accB
100101 accA ← accA ^ accB
101000 accA ← NOT(accA)
101001 accA ← accA AND accB
101010 accA ← accA OR accB
101011 accA ← accA XOR accB
101100 accA ← accA << accB
101101 accA ← accA <<< accB
101110 accA ← accA >> accB
101111 accA ← accA >>> accB
110000 accA CMP accB
CMP = 1
110001 isPrime(accA)
Prime = 1
110010 accB | accA
Divisible = 1
110011 rRAM[0x0E:0x0F] ← LCM
m= rRAM_adr[Operand [7:4]]
n= rRAM_adr[Operand [3:0]]
110100 Future 1
110101 Future 2
111111 STOP
valid =1
cc==16
yes
no
cc++
show result
done=1
flag
CMP
Prime
Divisible
accA > accB
accA < accB
accA == accB
gr=0
gr=1
ls=0
ls=1
eq=0
eq=1
CMP =0
Prime =0
Divisible =0
LCM
x = m
lcm = (m*n)/x
x == n
x > n
x = x-n
n = n-x

การออกแบบและพัฒนาส่วน Data Path





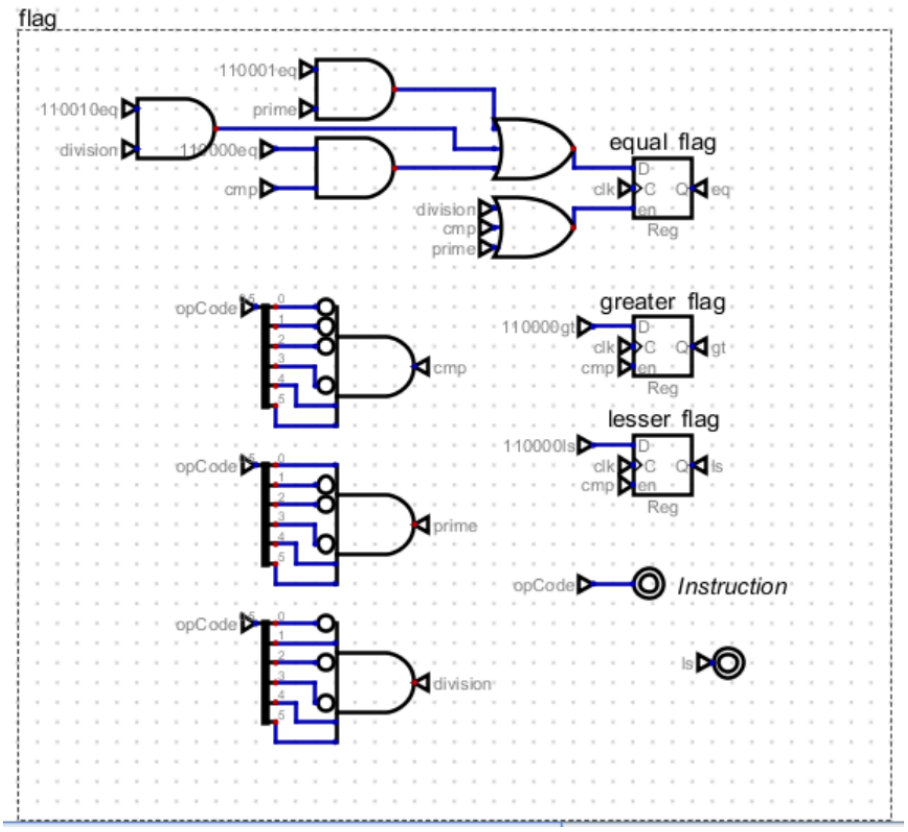
ข้อมูล 8 bits ที่ได้จากการ execute ของแต่ละ instruction จะออกมาจาก ALU ยกเว้น
- opcode ที่ต้อง jump จะทำใน mux อีกตัวเพื่อเลือกตามเงื่อนไขว่าจะ jump หรือไม่ คำสั่งที่ไม่เกี่ยวกับการ jump ให้ใส่ 0 ใน rom ใส่ค่าเป็น 1 ใน opcode ที่มีการ jump



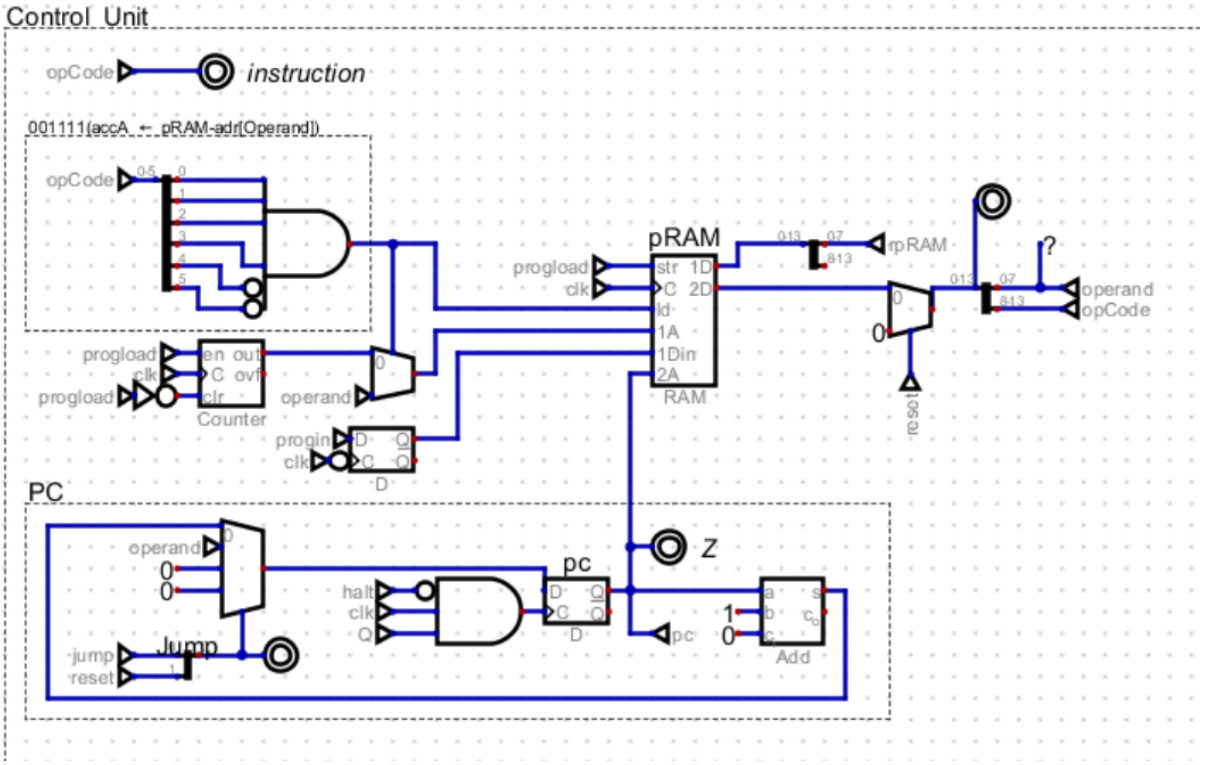
File

Address	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
16	0	0	1	1	1	1	1	1
24	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0

-ทำงานเกี่ยวกับ flag โดย ข้อมูลที่ได้จากการคำนวณจะเชื่อมกับ Register และ rRAM โดยตรง แต่ port str มีค่า 1 เข้าจึงจะสามารถเปลี่ยนแปลงค่าของ storage นั้นได



Control Unit



ในส่วนของ control unit จะมีองค์ประกอบหลักๆอยู่ สามส่วนคือ Program Counter (PC), Instruction Register (IR) และ Program RAM (pRAM) จะต้องทำให้ pc ใช้ clock น้อยที่สุดก่อนที่จะ fetch ค่าสั่งถัดไป ต่อมาจะเป็นรายละเอียดของ ส่วนประกอบของ control unit

PC

Program Counter เป็นรีจิสเตอร์ที่ใช้เก็บ **ที่อยู่ของคำสั่งถัดไป** หน้าที่หลักๆคือ ทำการ fetch ข้อมูลที่มีทั้ง operand และ opcode ซึ่งอยู่ใน pRAM ออก port 2D แล้วนำผลที่ได้ไปเก็บไว้ใน Instruction Register (IR)
โดยจะออกแบบให้ pc สามารถทำได้สามอย่างคือ
1 fetch คำสั่งถัดไป
2 jump ไปยัง address ต่างๆ
3 Reset เข้าสู่ Idle State ซึ่งในสภาวะนี้ PC จะไม่ทำการ fetch คำสั่งใหม่ (หยุดชั่วคราวเพื่อรอการเริ่มต้นระบบใหม่)ซึ่งเมื่อมีการ reset pc จะเข้าสู่ idle state หมายความว่าไม่ต้อง fetch ค่าสั่งถัดไปแล้ว

IR

ในส่วนนี้ จะกระจาย สัญญาณ 14 bit ที่ PC fetch มาแยกไปเป็น operand และ opcode แต่ไม่ใช้ clock เพื่อส่งต่อให้หน่วยควบคุม (control logic) และหน่วยปฏิบัติการ (ALU) ใช้งาน และในกรณีที่มีการกด reset จะทำการ เก็บค่า เป็น 0 (14 bits) แทน และ ไม่สนค่าที่ PC fetch มา

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 ค่าสั่งเข้า IR