

เอกสารด้านเทคนิคการออกแบบและพัฒนา

Digital Computer Logic Final Project : CPU

กลุ่ม : เหนียวไก่อ่ 40

สมาชิกกลุ่ม :

1. Pattadon Plubladpho
2. Pawat Lilapantisitti
3. Nattapat Yansungvon
4. Chayapol Champoonta

1. แนวคิดการออกแบบ CPU

ใช้หลักการ multicycle การทำงานเป็นดังนี้

Load: ใส่คำสั่ง binary ความยาว 14 bit ลงใน pRAM เก็บคำสั่งละ 1 ตำแหน่ง

Run: คำสั่งใน pRAM จะแบ่งเป็น 2 ส่วนได้แก่ Opcode ความยาว 6 bit(13-8) Operand ความยาว 8 bit(7-0)

ด้วยการเก็บไว้ใน ROM แล้วค่อย Decode แล้วรันคำสั่งตาม Opcode เข้า Exec_Mov, ALU, Exec_Jump, rRAM_write

Exec_Mov: ย้ายค่าต่างๆ เข้า AccA, AccB, RegC, RegD ทำตามคำสั่ง Opcode

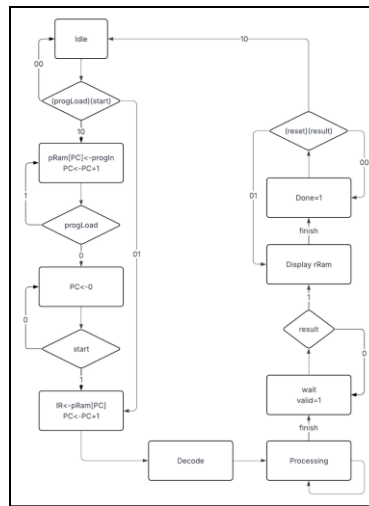
ALU: คำนวณค่าตามคำสั่ง Opcode แล้วพักค่าไว้ใน RegS

Exec_Jump: เมื่อคำสั่ง Opcode และเงื่อนไขตรง จะทำตามเงื่อนไข addr[Operand] ของ pRAM

rRAM_write: นำค่า AccA เข้า addr[Operand] ของ rRAM

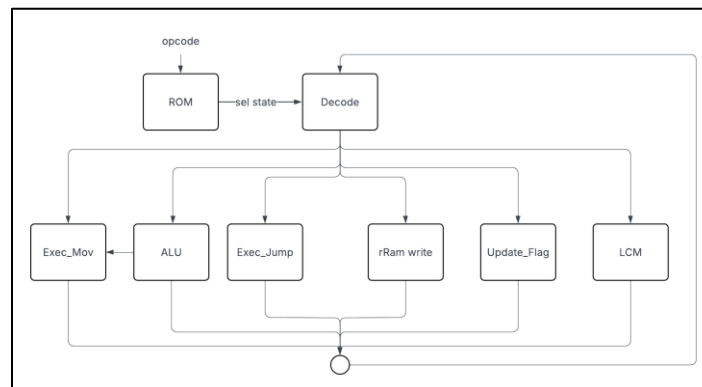
Result และ rRAM: จะประกอบไปด้วย component มากมาย ทั้งที่จากเก็บค่ามา และส่งค่าออก Output, ในส่วนของการ Reset rRAM เราจะใช้ 16 clock ในการลบค่าใน rRAM ทั้งหมด โดยเปลี่ยนค่าไปที่ละตำแหน่ง

2.ASM Chart / FSM Chart



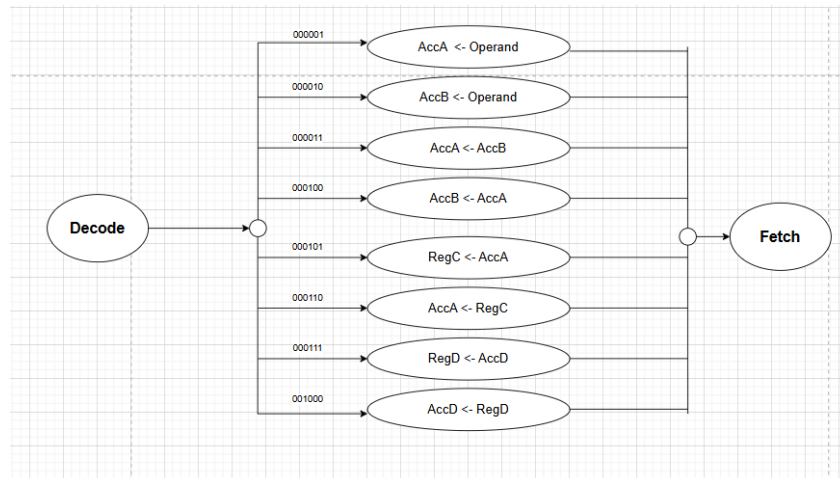
รูปที่ 1 - การทำงานภาพรวมของวงจร

เริ่มต้นที่ Idle รอจนค่า progload หรือค่า start เปลี่ยนค่าเป็น 1 หาก progload เป็น 1 ให้นำค่าจาก progin ไปเก็บไว้ใน pRam จนกว่า progload มีค่าเป็น 0 จากนั้นเมื่อ start มีค่า 1 นำข้อมูลจาก pRam เก็บใน IR จากนั้นเข้ากระบวนการ Decode และ Processing จนเสร็จกระบวนการให้แสดงค่า และเมื่อกด reset จะกลับไปยัง Idle

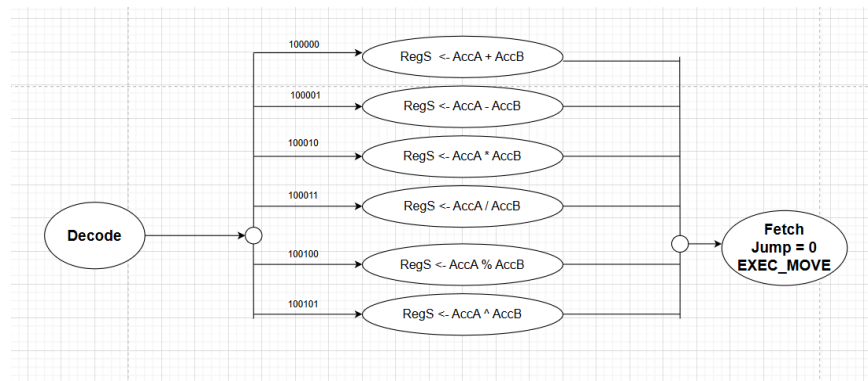


รูปที่ 2 แสดงการ decode ของวงจร

หลังจากที่ได้ input ค่า opcode มาแล้ว จะถูกส่งไปยัง ROM หลังจากนั้นจะทำการ Select State จาก ROM เพื่อทำการ decode ว่าใน state นี้ เราจะต้องไปทำ function ไหน ซึ่งจะมีเป็น 6 function คือ การเก็บและส่งต่อตัวแปร(EXEC_MOVE) , การคำนวณและ logic(ALU) , การส่งต่อข้อมูลไปยัง RAM เพื่อใช้ในการจดจำข้อมูล(rRAM write) , Update_Flag และการคำนวณ หรม. (LCM)



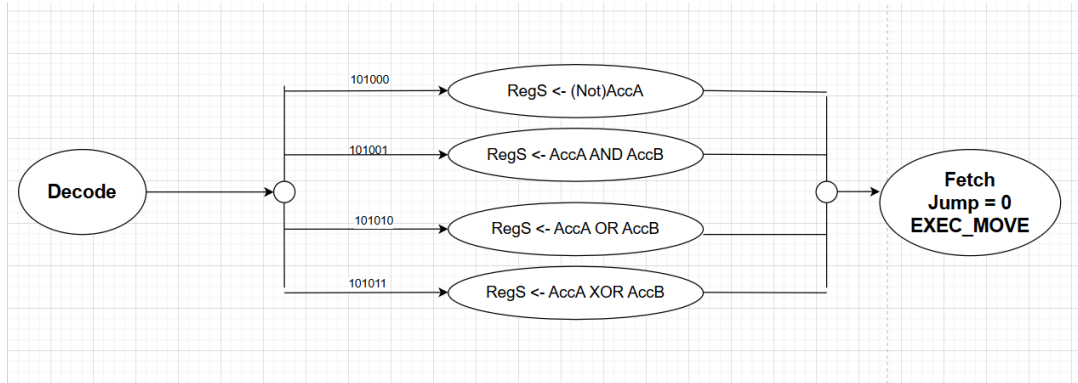
รูปที่ 3 - EXEC_MOVE



รูปที่ 4 - ALU

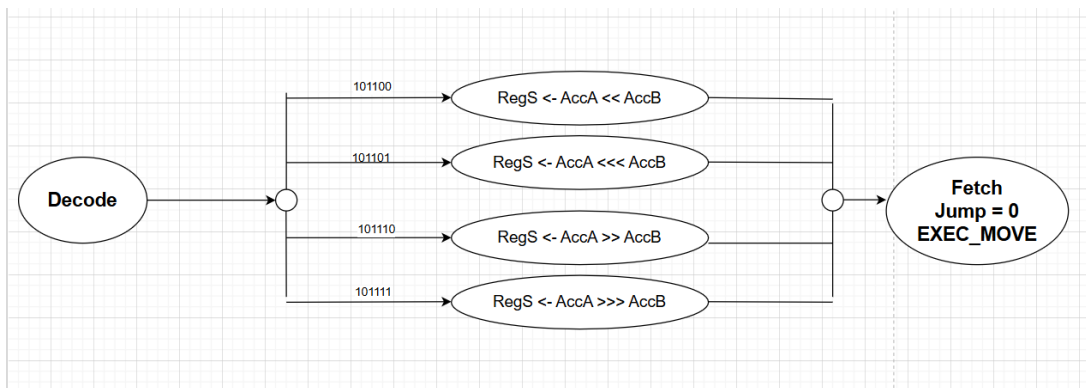
เป็นการทำตามคำสั่งที่ถูก Decode มา

โดยในส่วนนี้จะเป็คำสั่งที่เกี่ยวกับการดำเนินการทางคณิตศาสตร์ ได้แก่ การบวก, ลบ, คูณ, หาร, หารเอาเศษ, ยกกำลัง



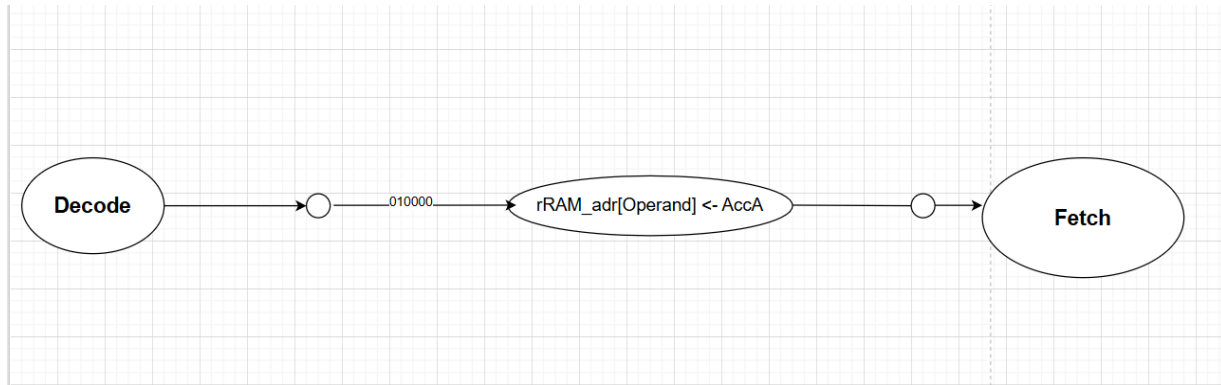
รูปที่ 5 – ALU

เป็นการทำตามคำสั่งที่ถูก Decode มา โดยในส่วนนี้จะเป็คำสั่งที่เป็นการดำเนินการกับ Bit ใน Operand ได้แก่ การกลับบิต, การ AND, การ OR, การ XOR



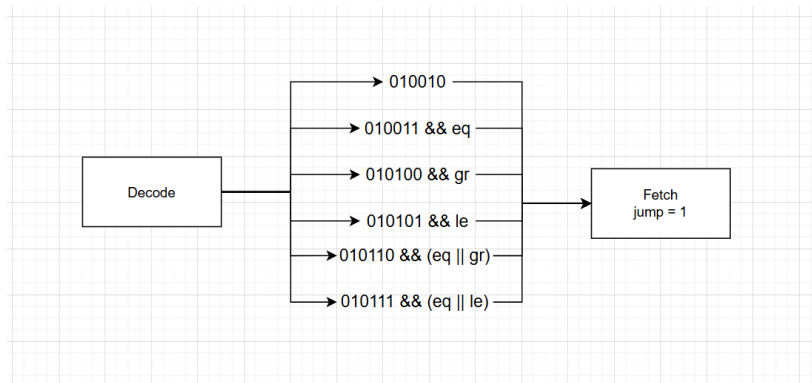
รูปที่ 6 – ALU

เป็นการทำตามคำสั่งที่ถูก Decode มา โดยในส่วนนี้จะเป็คำสั่งที่เป็นการดำเนินการกับ Bit ใน Operand ได้แก่ เลื่อนบิตไปทางซ้าย, หมุนบิตไปทางซ้าย, เลื่อนบิตไปทางขวา, หมุนบิตไปทางขวา



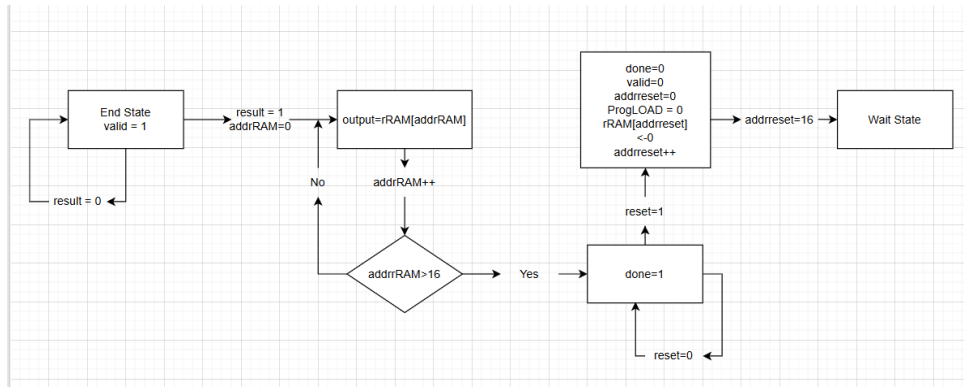
รูปที่ 7 – rRAM write

เป็นการทำตามคำสั่งที่ถูก decode มา โดยจะทำการนำค่าใน AccA ไปเก็บใน RAM ที่ Address ตาม Operand ในขณะนั้น



รูปที่ 8 – EXEC_JUMP

เป็นการทำตามคำสั่งที่ถูก decode ไว้ โดยจะทำการดึงค่า (fetch) และทำการกระโดดไปยังคำสั่งถัดไป (jump = 1) เมื่อเงื่อนไขต่างๆตามรูปเป็นจริง

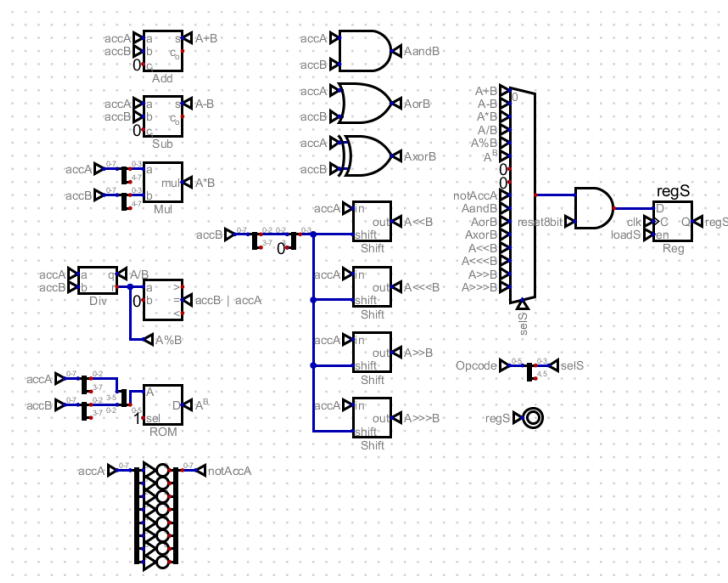


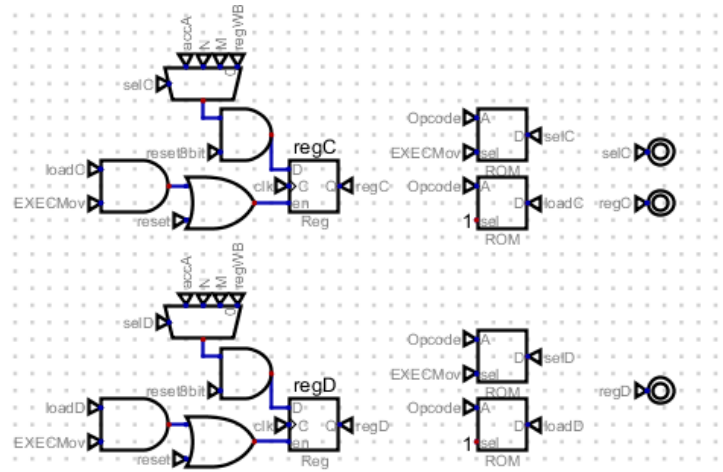
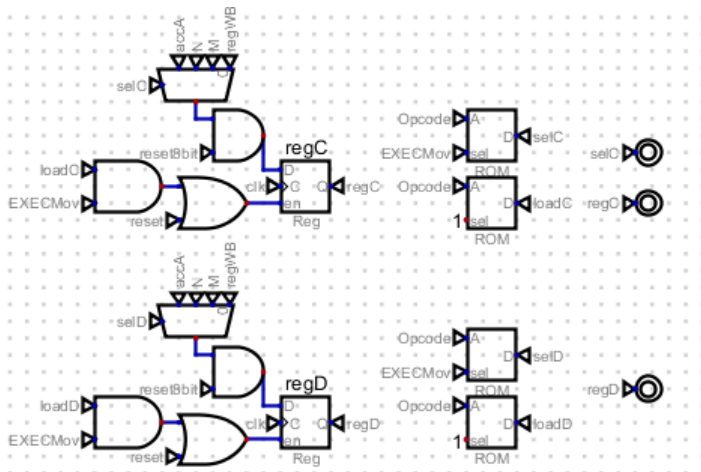
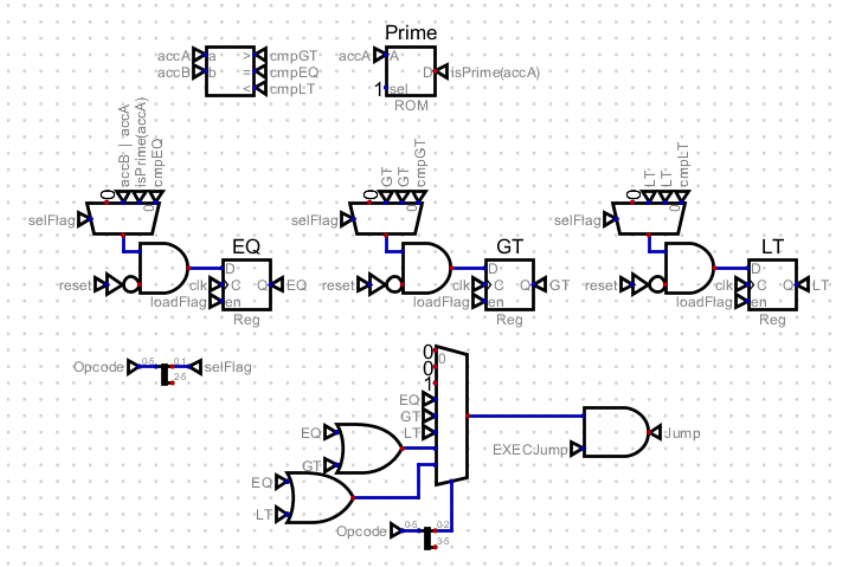
รูปที่ 9 – DISPLAY rRAM

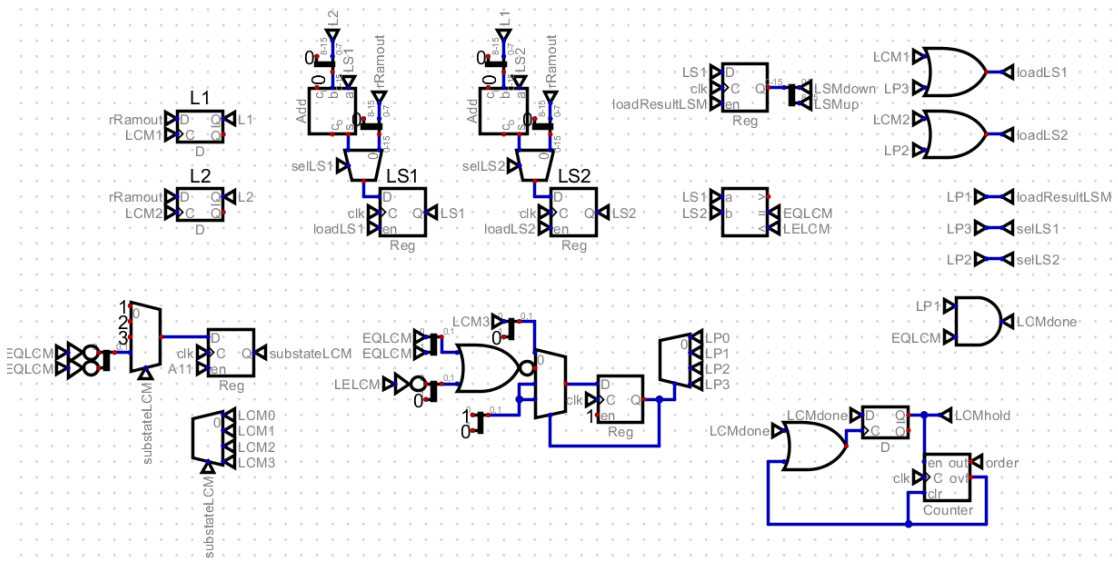
จะรอใน End State จน result เป็น 1 จะแสดงค่าผลลัพธ์จนครบ 16 ตัว จากนั้นให้ค่า done = 1 จากนั้นเมื่อเปลี่ยนค่า reset เป็น 1 จะทำการกลับ state เริ่มต้น แล้วเข้าสู่ Wait State

3.การออกแบบและพัฒนาส่วน Data Path

ALU







4.การออกแบบและพัฒนาส่วน Control Unit

Present State				Next state				Condition	next state name
A	B	C	D	A+	B+	C+	D+		
0	0	0	0	0	0	0	0	Proglod = 0	Idle
				0	0	0	1	Proglod = 1	proglin->pRam
0	0	0	1	0	0	0	1	Proglod = 1	proglin->pRam
				0	0	1	0	Proglod = 0	wait start
0	0	1	0	0	0	1	0	Start = 0	wait start
				0	0	1	1	Start = 1	Fetch

Present State				Next state				Condition	next state name
A	B	C	D	A+	B+	C+	D+		
0	0	1	1	0	1	0	0	None	Decode
0	1	0	0	0	0	0	0	Opcode = 111111	Idle
				0	0	1	1	Opcode = 000000	Fetch

				0	1	0	1	Opcode = 000001-010000	Exec_Move
				0	1	1	0	Opcode = 010010-010111	Exec_Jump
				0	1	1	1	Opcode = 010001	rRAM_Write

Present State				Next state				Condition	next state name
A	B	C	D	A+	B+	C+	D+		
0	1	0	0	1	0	0	0	Opcode = 100000-101111	ALU
				1	0	1	0	Opcode = 110000-110010	Update_Flag
				1	0	1	1	Opcode = 110011	LCM

Present State				Next state				Condition	next state name
A	B	C	D	A+	B+	C+	D+		
0	1	0	0	1	0	0	0	Opcode = 100000-101111	ALU
				1	0	1	0	Opcode = 110000-110010	Update_Flag
				1	0	1	1	Opcode = 110011	LCM

Circuit

