อธิบายความหมายของตารางที่ 2.3

คำสั่งทุกคำสั่งจะจัดการทำงานเป็น 5 จังหวะ แต่ละจังหวะใช้สัญญาณนาฬิกา 1 ไซเคิล ดังนั้นคำสั่งแต่ละคำสั่งใช้เวลาทำงานเท่ากับ 5 คาบเวลาของสัญญาณนาฬิกา ความหมายของคำสั่งที่เขียนไว้ในตารางที่ 2.3 มีดังนี้

คำสั่งรหัส 1 หมายถึงคำสั่งให้เก็บข้อมูล n ขนาด 4 บิตไว้ที่รีจิสเตอร์ A มี รูปแบบคำสั่ง MOV A,#n มีการทำงานในแต่ละจังหวะดังนี้

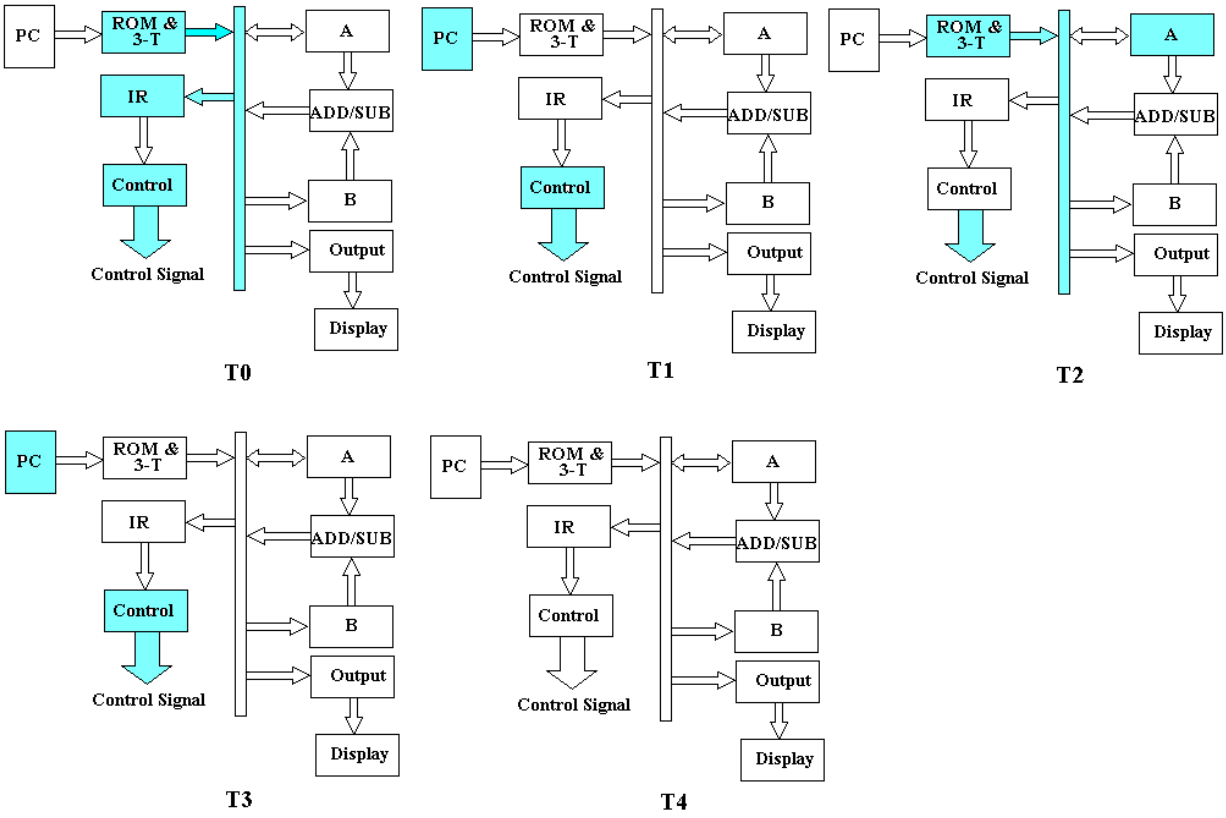
จังหวะแรก T0 IR <- Memory หมายถึง นำคำสั่งที่อยู่ในหน่วยความจำมาเก็บที่รีจิสเตอร์คำสั่ง IR

จังหวะ T1 PC <- PC+1 หมายถึง เพิ่มค่า PC ขึ้น 1

จังหวะ T2 A <- Memory หมายถึง นำข้อมูลที่อยู่ในหน่วยความจำมาเก็บที่รีจิสเตอร์คำสั่ง A

จังหวะ T3 PC <- PC+1 หมายถึง เพิ่มค่า PC ขึ้น 1

จังหวะ T4 หมายถึง ไม่มีการทำงานใดๆ ปล่อยให้สัญญาณนาฬิกาผ่านไป 1 ไซเคิล



รูปที่ 2.17 การทำงานของคำสั่ง MOV A,#n

คำสั่งรหัส 2 หมายถึงคำสั่งให้บวกข้อมูลในรีจิสเตอร์ A เข้ากับข้อมูล n ผลลัพธ์เก็บที่รีจิสเตอร์ A มีรูปแบบคำสั่ง ADD A,#n มีการทำงานในแต่ละจังหวะดังนี้

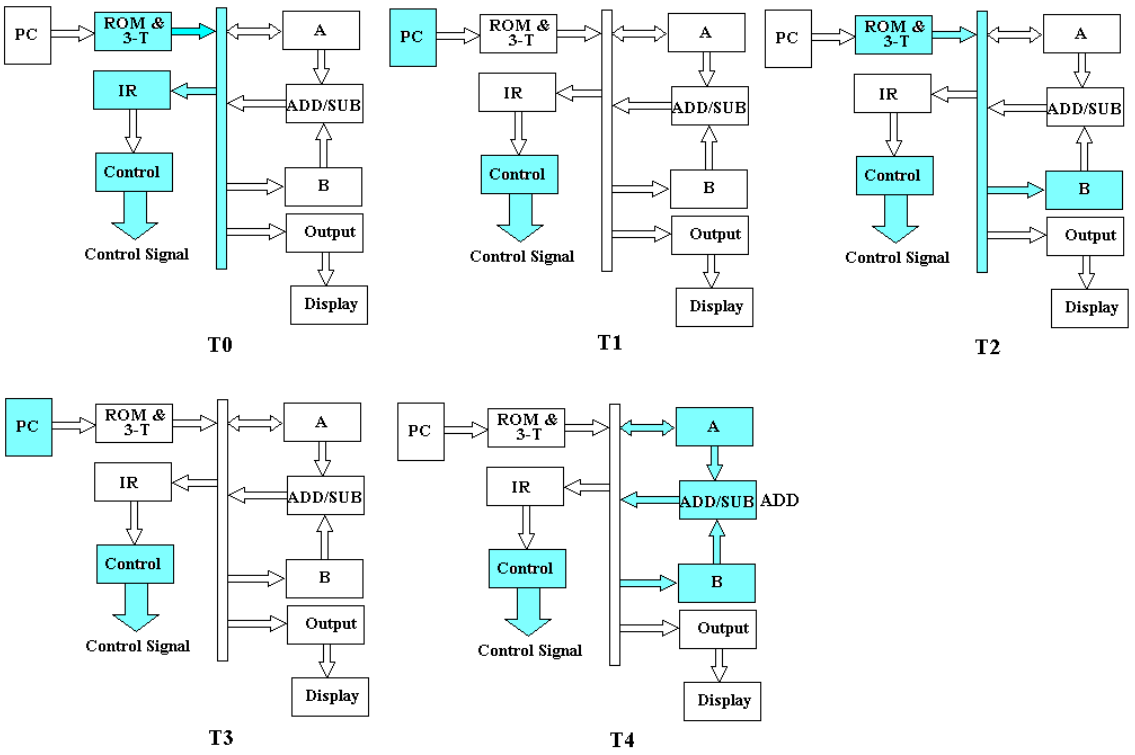
จังหวะแรก T0 IR <- Memory หมายถึง นำคำสั่งที่อยู่ในหน่วยความจำมาเก็บที่รีจิสเตอร์คำสั่ง IR

จังหวะ T1 PC <- PC+1 หมายถึง เพิ่มค่า PC ขึ้น 1

จังหวะ T2 B <- Memory หมายถึง นำข้อมูลที่อยู่ในหน่วยความจำมาเก็บที่รีจิสเตอร์คำสั่ง B

จังหวะ T3 PC <- PC+1 หมายถึง เพิ่มค่า PC ขึ้น 1

จังหวะ T4 A <- A + B หมายถึง บวกข้อมูลในรีจิสเตอร์ A เข้ากับข้อมูลในรีจิสเตอร์ B ผลลัพธ์เก็บที่รีจิสเตอร์ A



รูปที่ 2.18 การทำงานของคำสั่ง ADD A,#n

คำสั่งรหัส 3 หมายถึงคำสั่งให้ลบข้อมูลในรีจิสเตอร์ A ด้วยข้อมูล n ผลลัพธ์เก็บที่รีจิสเตอร์ A มีรูปแบบคำสั่ง SUB A,#n มีการทำงานในแต่ละจังหวะดังนี้

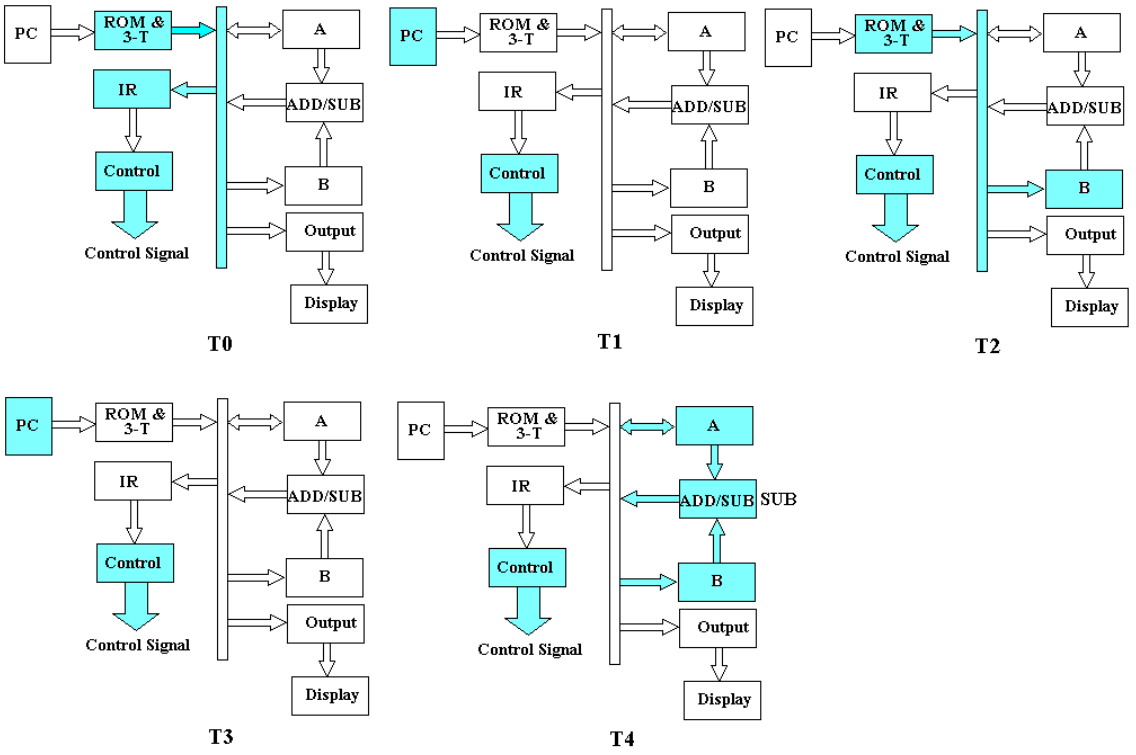
จังหวะแรก T0 IR <- Memory หมายถึง นำคำสั่งที่อยู่ในหน่วยความจำมาเก็บที่รีจิสเตอร์คำสั่ง IR

จังหวะ T1 PC <- PC+1 หมายถึง เพิ่มค่า PC ขึ้น 1

จังหวะ T2 B <- Memory หมายถึง นำข้อมูลที่อยู่ในหน่วยความจำมาเก็บที่รีจิสเตอร์คำสั่ง B

จังหวะ T3 PC <- PC+1 หมายถึง เพิ่มค่า PC ขึ้น 1

จังหวะ T4 A <- A - B หมายถึง ลบข้อมูลในรีจิสเตอร์ A ด้วยข้อมูลในรีจิสเตอร์ B ผลลัพธ์เก็บที่รีจิสเตอร์ A



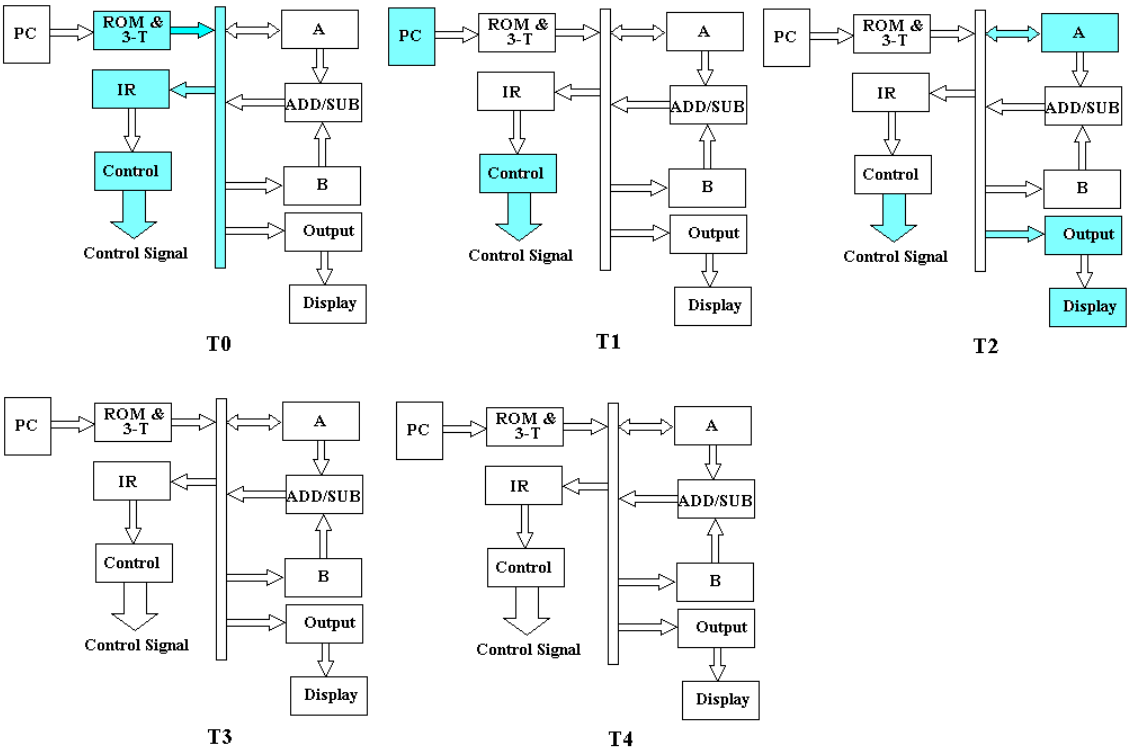
รูปที่ 2.19 การทำงานของคำสั่ง SUB A,#n

คำสั่งรหัส 4 หมายถึงคำสั่งให้ส่งข้อมูลในรีจิสเตอร์ A ออกไปที่เอาท์พุท มีรูปแบบคำสั่ง OUT มีการทำงานในแต่ละจังหวะดังนี้

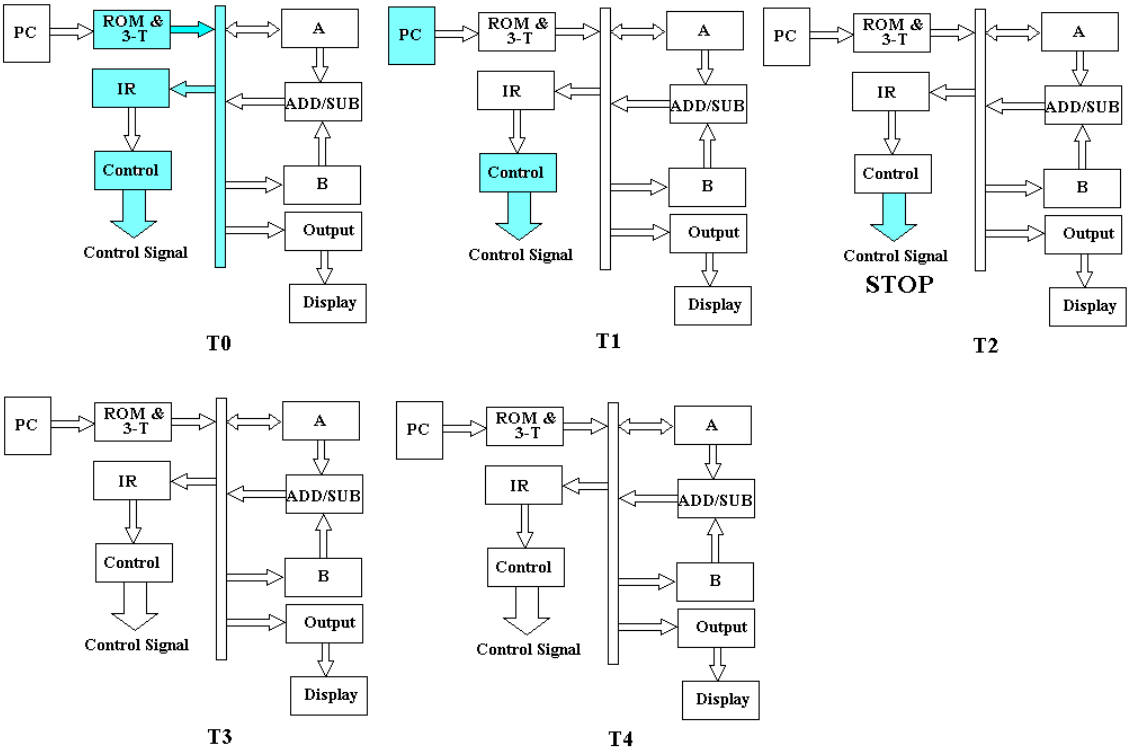
จังหวะแรก T0	IR <- Memory	หมายถึง นำคำสั่งที่อยู่ในหน่วยความจำมาเก็บที่รีจิสเตอร์คำสั่ง IR
จังหวะ T1	PC <- PC+1	หมายถึง เพิ่มค่า PC ขึ้น 1
จังหวะ T2	O/P <- A	หมายถึง นำข้อมูลที่อยู่ในรีจิสเตอร์คำสั่ง A ส่งให้รีจิสเตอร์เอาท์พุท
จังหวะ T3		หมายถึง ไม่มีการทำงานใดๆ ปล่อยให้สัญญาณนาฬิกาผ่านไป 1 ไซเคิล
จังหวะ T4		หมายถึง ไม่มีการทำงานใดๆ ปล่อยให้สัญญาณนาฬิกาผ่านไป 1 ไซเคิล

คำสั่งรหัส 5 หมายถึงคำสั่งให้หยุดการทำงาน มี รูปแบบคำสั่ง HALT มีการทำงานในแต่ละจังหวะดังนี้

จังหวะแรก T0	IR <- Memory	หมายถึง นำคำสั่งที่อยู่ในหน่วยความจำมาเก็บที่รีจิสเตอร์คำสั่ง IR
จังหวะ T1	PC <- PC+1	หมายถึง เพิ่มค่า PC ขึ้น 1
จังหวะ T2	STOP	หมายถึง หยุดการทำงาน ในที่นี้ใช้วิธีหยุดจ่ายสัญญาณนาฬิกา
จังหวะ T3		หมายถึง ไม่มีการทำงานใดๆ
จังหวะ T4		หมายถึง ไม่มีการทำงานใดๆ



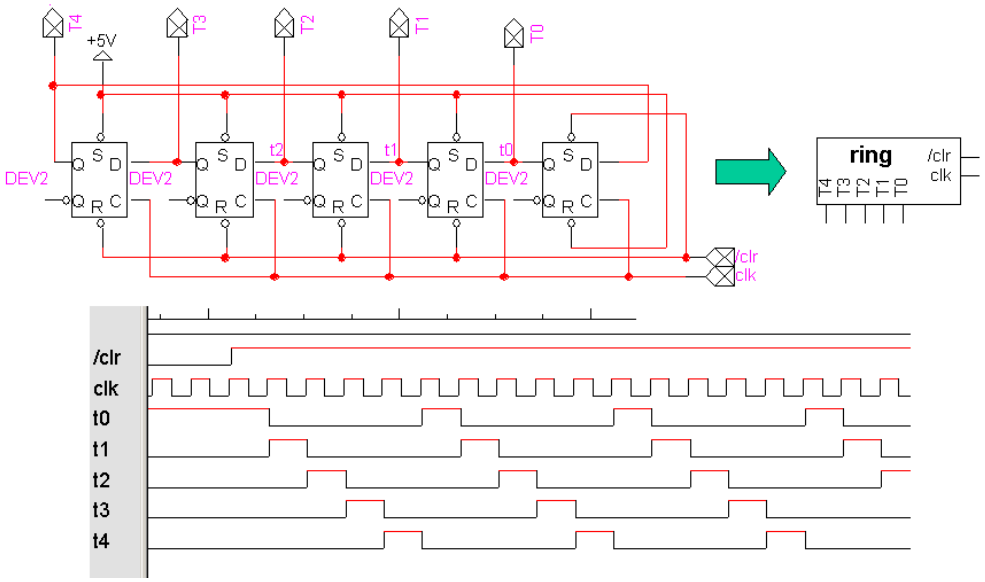
รูปที่ 2.20 การทำงานของคำสั่ง OUT



รูปที่ 2.21 การทำงานของคำสั่ง HALT

2.11 วงจรนับแบบวงแหวน (Ring Counter)

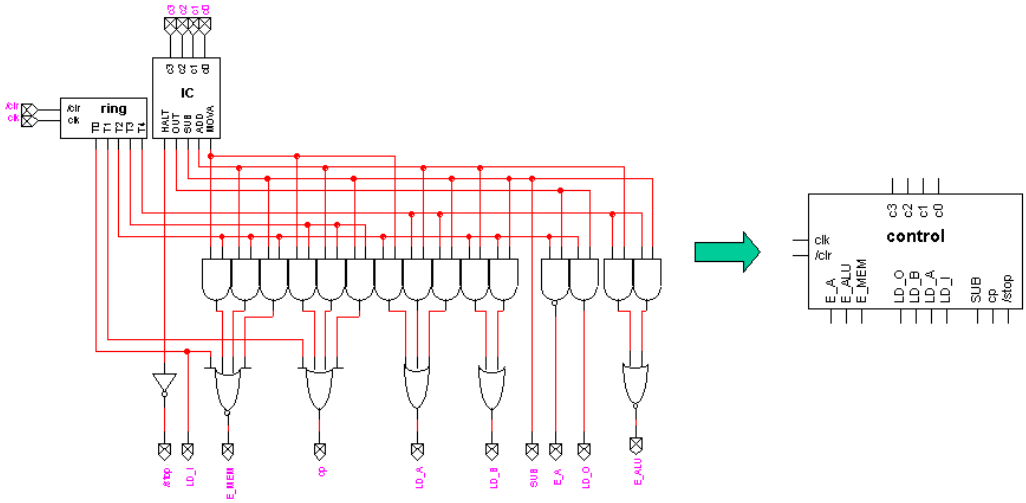
วงจรนับแบบวงแหวนมีลักษณะการทำงานที่ในแต่ละเวลาจะให้สัญญาณเอาท์พุทเป็นโลจิก 1 ได้ครั้งละ 1 บิตเท่านั้น ดังนั้นจึงใช้วงจรนับแบบนี้เป็นหน่วยกำหนดจังหวะการทำงาน โดยถ้าบิตเอาท์พุท T0 เป็น 1 หมายถึงขณะนี้เป็นจังหวะ T0 บิตเอาท์พุทอื่นๆก็เป็นไปในทำนองเดียวกัน



รูปที่ 2.22 โลจิกไดอะแกรมและสัญญลักษณ์ของวงจรนับแบบวงแหวน

2.12 หน่วยควบคุม (Control Unit)

หน่วยควบคุมทำหน้าที่ควบคุมการทำงานของระบบให้เป็นไปตามคำสั่งต่างๆที่ออกแบบไว้ วงจรควบคุมประกอบด้วย วงจรนับแบบวงแหวน วงจรถอดรหัสคำสั่ง และวงจรเกตต่างๆที่ทำหน้าที่ถอดรหัสคำสังและจังหวะการทำงานที่ได้จากวงจรนับแบบวงแหวน แล้วสร้างเป็นสัญญาณควบคุมส่งไปยังหน่วยต่างๆภายในระบบ ตามตารางที่ 2.4



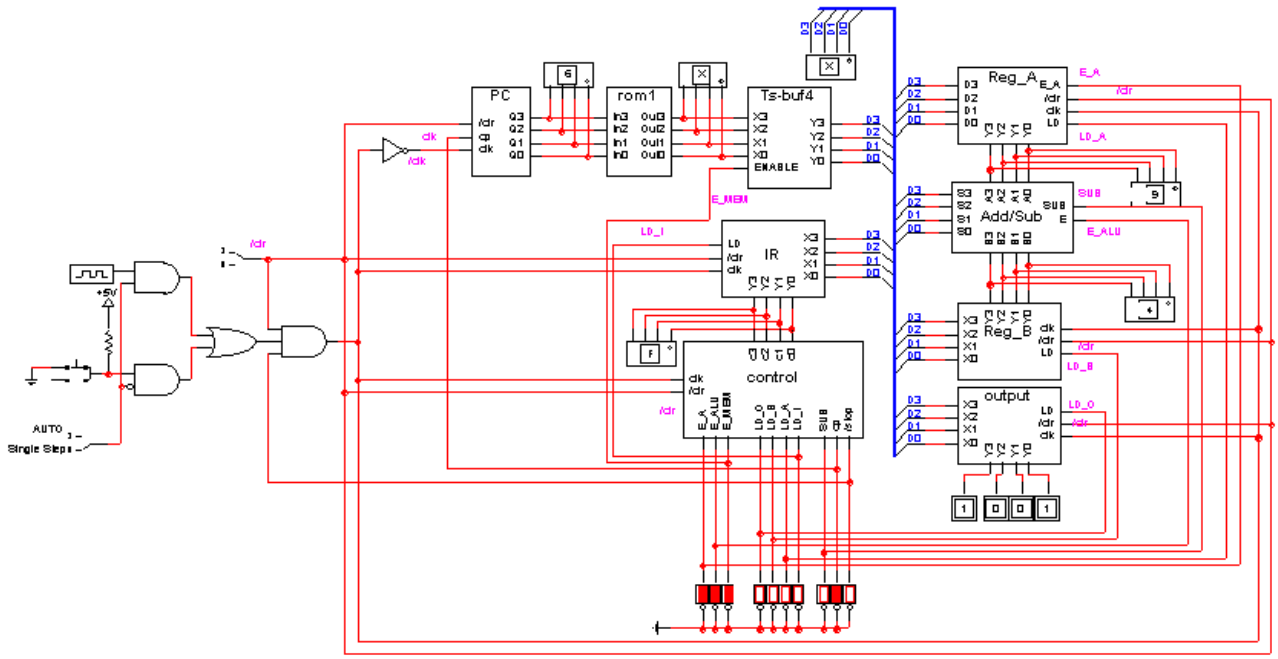
รูปที่ 2.23 โลจิกไดอะแกรมและสัญญลักษณ์ของหน่วยควบคุม

ตารางที่ 2.4 สัญญาณที่ทำงานในแต่ละจังหวะ

		LD_I	LD_A	LD_B	LD_O	E_MEM	E_A	E_ALU	CP	SUB	STOP	ความหมาย
MOV A,#n	T0	1	0	0	0	0	1	1	0	0	1	IR <- Memory
	T1	0	0	0	0	1	1	1	1	0	1	PC <- PC+1
	T2	0	1	0	0	0	1	1	0	0	1	A <- Memory
	T3	0	0	0	0	1	1	1	1	0	1	PC <- PC+1
	T4	0	0	0	0	1	1	1	0	0	1	
ADD A,#n	T0	1	0	0	0	0	1	1	0	0	1	IR <- Memory
	T1	0	0	0	0	1	1	1	1	0	1	PC <- PC+1
	T2	0	0	1	0	0	1	1	0	0	1	B <- Memory
	T3	0	0	0	0	1	1	1	1	0	1	PC <- PC+1
	T4	0	1	0	0	1	1	0	0	0	1	A <- A + B
SUB A,#n	T0	1	0	0	0	0	1	1	0	0	1	IR <- Memory
	T1	0	0	0	0	1	1	1	1	0	1	PC <- PC+1
	T2	0	0	1	0	0	1	1	0	0	1	B <- Memory
	T3	0	0	0	0	1	1	1	1	1	1	PC <- PC+1
	T4	0	1	0	0	1	1	0	0	1	1	A <- A - B
OUT	T0	1	0	0	0	0	1	1	0	0	1	IR <- Memory
	T1	0	0	0	0	1	1	1	1	0	1	PC <- PC+1
	T2	0	0	0	1	1	0	1	0	0	1	O/P <- A
	T3	0	0	0	0	1	1	1	0	0	1	
	T4	0	0	0	0	1	1	1	0	0	1	
HALT	T0	1	0	0	0	0	1	1	0	0	1	IR <- Memory
	T1	0	0	0	0	1	1	1	1	0	1	PC <- PC+1
	T2	0	0	0	0	1	1	1	0	0	0	STOP
	T3	0	0	0	0	1	1	1	0	0	0	
	T4	0	0	0	0	1	1	1	0	0	0	

2.13 สถาปัตยกรรมและบล็อกไดอะแกรมของ SiCo

เมื่อนำหน่วยต่างๆมาประกอบรวมกันได้สถาปัตยกรรมของ SiCo ตามรูปที่ 2.19 และบล็อกไดอะแกรมของระบบแสดงอยู่ในรูปที่ 2.15 การทำงานของระบบทำได้ทั้งแบบอัตโนมัติคือทำตั้งแต่คำสั่งแรกจนสินสุดคำสั่งสุดท้าย หรือทำแบบที่ละคำสั่ง โดยการเลือกวิธีการทำงานด้วยสวิตช์ Auto/ Single Step



รูปที่ 2.24 สถาปัตยกรรม SiCo

ตัวอย่างโปรแกรม

เป็นคำสั่งบวกเลข 5 เข้ากับเลข 4 แล้วนำคำตอบออกแสดงผลที่เอาท์พุท โปรแกรมสามารถเขียนได้ดังนี้

		Mnemonic		
Address	Data	Opcode	Operand	Comment
		ORG	0	
0	1 5	MOV	A,#5	; Set A = 5
2	2 4	ADD	A,#4	; Add A with 4
4	4	OUT		; Display result
5	F	HALT		; Stop

โปรแกรมจัดเก็บลงในหน่วยความจำตามตำแหน่งต่างๆดังนี้

หน่วยความจำ

Address	Data
0	1
1	5
2	2
3	4
4	4
5	F

2.14 Microprogramming

จากที่กล่าวมาแล้วในหัวข้อที่ 2.12 ในหน่วยควบคุมจะใช้วงจรเกตเพื่อสร้างสัญญาณควบคุมต่างๆ การทำลักษณะนี้ไม่มีความยืดหยุ่น การปรับแต่งการทำงานทำได้ยากเพราะว่าต้องแก้ไขวงจรใหม่ อีกวิธีหนึ่งที่นิยมใช้กันมากกว่าคือการใช้ หน่วยความจำ ROM มาแทนวงจรเกต โดยให้บัสแอดเดรสแทนสัญญาณอินพุทของวงจรเกตและ บัสข้อมูลเป็นสัญญาณควบคุมต่างๆ เมื่อต้องการให้สัญญาณควบคุมเป็นอะไรเมื่อสัญญาณอินพุทเป็นอย่างไรก็ใช้การบันทึกเก็บไว้ใน ROM ตัวอย่างเช่น คำสั่ง MOV A,#n จากตารางที่ 2.4 เมื่อจังหวะ T0 สัญญาณควบคุมเป็นดังนี้

		LD_I	LD_A	LD_B	LD_O	E_MEM	E_A	E_ALU	CP	SUB	STOP
MOV A,#n	T0	1	0	0	0	0	1	1	0	0	1

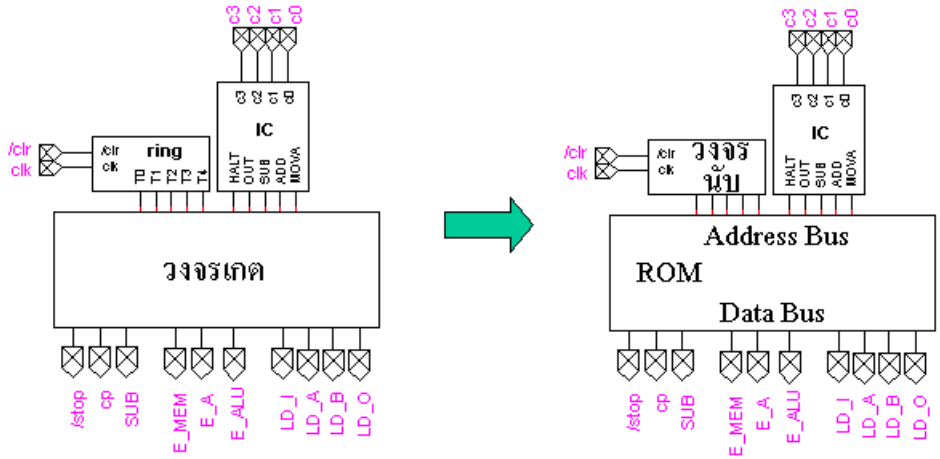
เมื่อเขียนเป็นค่าโลจิกต่างๆได้เป็น

T0	T1	T2	T3	T4	HALT	OUT	SUB	ADD	MOV	LD_I	LD_A	LD_B	LD_O	E_MEM	E_A	E_ALU	CP	SUB	STOP
1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1

แปลงเป็นสัญญาณสำหรับ ROM

Address Bus										Data Bus									
A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1

หมายความว่าที่ตำแหน่ง 1000000001B ของ ROM จะต้องบันทึกข้อมูลเป็น 1000011001B การทำเช่นนี้เรียกว่า " Microprogramming" การใช้สัญญาณจาก ROM มาทำหน้าที่แทนวงจรเกตทำให้สามารถแก้ไขและเปลี่ยนแปลงการทำงานของวงจรได้ง่ายขึ้น



รูปที่ 2.25 ลักษณะการใช้ Microprogramming แทนวงจรเกต