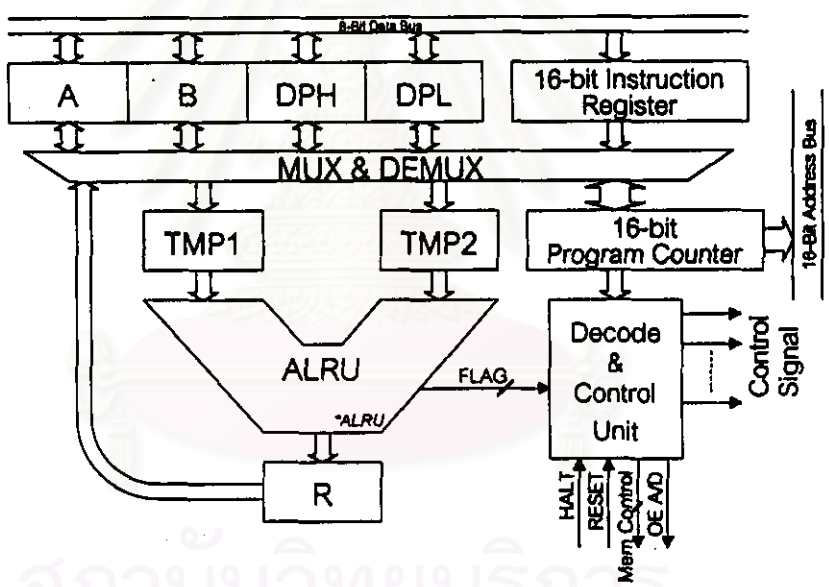
บทที่ 5

หน่วยประมวลผลกลาง

5.1 โครงสร้างภายในของหน่วยประมวลผลกลาง

หน่วยประมวลผลกลางทำหน้าที่ควบคุมการทำงานของส่วนต่างๆ ภายในทีวีไมโครคอนโทรลเลอร์ โดยจะทำงานตามคำสั่งซึ่งเก็บไว้ในหน่วยความจำอ่านอย่างเดียว โครงสร้างภายในหน่วยประมวลผลกลางแสดงดังรูปที่ 5.1 ประกอบด้วยส่วนต่างๆ ที่มีหน้าที่ดังนี้



รูปที่ 5.1 โครงสร้างภายในของหน่วยประมวลผลกลาง

1. หน่วยคำนวณ, ตรรก และหมุน (Arithmetic Logic Rotate Unit : ALRU) เป็นส่วนที่ใช้ในการคำนวณทางคณิตศาสตร์, การดำเนินการทางตรรก และการหมุนข้อมูล โดยรับข้อมูลมาทางรีจิสเตอร์ TMP1 กับ TMP2 และให้ผลลัพธ์ที่รีจิสเตอร์ R

2. รีจิสเตอร์คำสั่ง (Instruction Register) เป็นรีจิสเตอร์ขนาด 16 บิต ใช้เก็บคำสั่งที่อ่านมาจากบัสข้อมูล (Data Bus) ขนาด 8 บิต 2 ครั้ง เพื่อส่งไปยังหน่วยถอดรหัสและควบคุมต่อไป

โดยถ้าเป็นคำสั่งที่ดำเนินการระหว่างรีจิสเตอร์กับค่าคงที่จะส่งค่าคงที่จากคำสั่งนั้นไปยังหน่วยคำนวณ, ตรรก และหมุน เพื่อประมวลผล

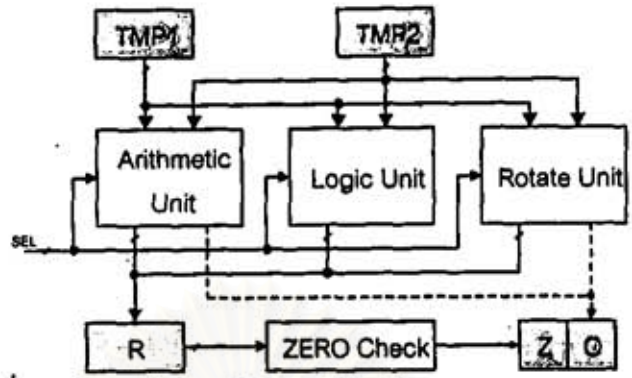
3. หน่วยถอดรหัสและควบคุม (Decode and Control Unit) เป็นส่วนที่ถอดรหัสจากรีจิสเตอร์คำสั่ง และควบคุมการทำงานในส่วนต่างๆ คือ ควบคุมการอ่าน, การเขียนข้อมูล ทั้งภายในและภายนอกหน่วยประมวลผลกลาง โดยจะสั่งให้หยุดการทำงานของหน่วยประมวลผลกลางชั่วขณะหากมีสัญญาณ Halt จากภายนอกเข้ามา

4. รีจิสเตอร์ (Register) ประกอบไปด้วยรีจิสเตอร์ขนาด 8 บิต ที่สามารถใช้งานได้จำนวน 4 รีจิสเตอร์ คือ รีจิสเตอร์ A, B, DPL และ DPH ซึ่ง 2 รีจิสเตอร์คือ DPL (Data Pointer Low byte) และ DPH (Data Pointer High byte) จะสามารถนำมาเรียงต่อกันเป็น รีจิสเตอร์ขนาด 16 บิต เพื่อใช้ในการอ้างอิงตำแหน่งของหน่วยความจำสำหรับอ่านข้อมูล, เขียนข้อมูล หรือกระโดดได้

5. โปรแกรมเคาน์เตอร์ (Program Counter : PC) เป็นรีจิสเตอร์ขนาด 16 บิต เพื่อใช้อ้างอิงตำแหน่งของโปรแกรมที่เก็บไว้ในหน่วยความจำผ่านทางบัสตำแหน่งขนาด 16 บิต โดยค่าของโปรแกรมเคาน์เตอร์จะถูกกำหนดจากรีจิสเตอร์ DPH และ DPL หรือได้รับการคำนวณจากหน่วยคำนวณ, ตรรก และหมุน ซึ่งขึ้นอยู่กับชุดคำสั่งที่ได้รับ

5.2 หน่วยคำนวณ, ตรรก และหมุน (Arithmetic Logic Rotate Unit: ALRU)

หน่วยคำนวณ, ตรรก และหมุนเป็นส่วนที่ใช้ดำเนินการ (Operation) ของหน่วยประมวลผลกลางซึ่งมีโครงสร้างภายใน ดังรูปที่ 5.2 ประกอบไปด้วยส่วนที่ทำงานพร้อมๆ กัน 3 ส่วน คือ หน่วยคำนวณ (Arithmetic Unit) ทำหน้าที่คำนวณทางคณิตศาสตร์ คือ บวก, ลบ และโอนย้ายข้อมูล, หน่วยตรรก (Logic Unit) ทำหน้าที่ดำเนินการทางตรรก คือ and, or และ xor และหน่วยหมุน (Rotate Unit) ทำหน้าที่หมุน (Rotate) ข้อมูลทางซ้าย และทางขวา ส่วนสัญญาณ SEL ทำหน้าที่เลือกวิธีดำเนินการ ซึ่งจะเป็นไปตามตารางรูปที่ 5.3 และผลลัพธ์ที่ได้จากหน่วยคำนวณ, ตรรก และหมุน จะได้รับการตรวจค่าว่าเป็นศูนย์หรือไม่ด้วยวงจรตรวจสอบค่าศูนย์ (ZERO Check) เพื่อกำหนดค่าให้กับแฟล็กศูนย์ (Z) ส่วนแฟล็กตัวทด (C) ได้มาจากหน่วยคำนวณ หรือหน่วยหมุน



รูปที่ 5.2 โครงสร้างภายในของหน่วยคำนวณ, ตรรก และหมุน

SEL [3...2]	SEL [1...0]			
	00	01	10	11
00	TMP1 – TMP2	TMP1 + TMP2	TMP1 – TMP2 - C	TMP1 + TMP2 + C
01	TMP1 and TMP2	TMP1 or TMP2	TMP1 xor TMP2	-
10	Rotate TMP1 Left	Rotate TMP1 Left with C	Rotate TMP1 Right	Rotate TMP1 Right with C
11	-	-	-	-

รูปที่ 5.3 ตารางแสดงการดำเนินการของหน่วยคำนวณ, ตรรก และหมุน

5.3 ชุดคำสั่งของหน่วยประมวลผลกลาง

หน่วยประมวลผลกลางมีคำสั่งทั้งหมด 35 คำสั่ง แสดงดังรูปที่ 5.4 ซึ่ง Reg หมายถึง รีจิสเตอร์ A, B, C, DPL และ DPH ส่วน Acc หมายถึง รีจิสเตอร์ A และ B เท่านั้น

Instruction	Meaning
Sub Reg,#Const	Subtract register with constant
Sub Reg,Reg	Subtract register with register
Add Reg,#Const	Add register with constant
Add Reg,Reg	Add register with register
Suc Reg,#Const	Subtract register with constant and carry
Suc Reg,Reg	Subtract register with register and carry
Adc Reg,#Const	Add register with constant and carry
Adc Reg,Reg	Add register with register and carry

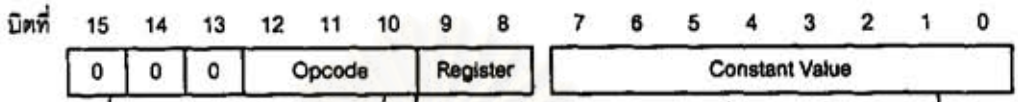
รูปที่ 5.4 ตารางแสดงชุดคำสั่งของหน่วยประมวลผลกลาง

Instruction	Meaning
And Reg,#Const	Perform AND operation between register and constant
And Reg,Reg	Perform AND operation between register and register
Or Reg,#Const	Perform OR operation between register and constant
Or Reg,Reg	Perform OR operation between register and register
Xor Reg,#Const	Perform XOR operation between register and constant
Xor Reg,Reg	Perform XOR operation between register and register
Mov Reg,#Const	Move constant to register
Mov Reg,Reg	Move register to register
Rol Reg	Rotate register left
Ror Reg	Rotate register right
Rlc Reg	Rotate register left with carry
Rrc Reg	Rotate register right with carry
Ld_pcl Acc	Load accumulator from program counter low byte
Ld_pch Acc	Load accumulator from program counter high byte
Lod Acc	Load accumulator from address DPH:DPL
Lod Acc,@	Load accumulator from address 80:@
Lod Acc,MAP	Load accumulator from memory mapped I/O
Ld_adc Acc	Load accumulator from A/D
Sto Acc	Store accumulator to address DPH:DPL
Sto @,Acc	Store accumulator to address 80:@
Sto MAP,Acc	Store accumulator to memory mapped I/O
Jnc Rel 9 bits	Jump when carry flag is cleared to 9 bits relative address
Jc Rel 9 bits	Jump when carry flag is set to 9 bits relative address
Jnz Rel 9 bits	Jump when zero flag is cleared to 9 bits relative address
Jz Rel 9 bits	Jump when zero flag is set to 9 bits relative address
Jr Rel 9 bits	Jump to 9 bits relative address
Jmp	Jump to address DPH:DPL

รูปที่ 5.4 (ต่อ) ตารางแสดงชุดคำสั่งของหน่วยประมวลผลกลาง

ทั้ง 35 คำสั่งนี้สามารถแบ่งออกได้เป็น 5 ประเภท คือ คำสั่งดำเนินการระหว่างรีจิสเตอร์กับค่าคงที่, คำสั่งดำเนินการระหว่างรีจิสเตอร์อย่างเดียว, คำสั่งโหลดค่าโปรแกรมเคาท์เตอร์, คำสั่ง Load หรือ Store และคำสั่งกระโดด โดยมีรายละเอียดดังนี้

1. การดำเนินการระหว่างรีจิสเตอร์กับค่าคงที่ ผลลัพธ์ที่ได้จะเก็บค่าไว้ที่รีจิสเตอร์ คำสั่งที่อยู่ในกลุ่มนี้ ได้แก่ Sub, Add, Suc, Adc, And, Or, Xor และ Mov ซึ่งจะมีลักษณะการใช้งาน คือ Inst. Reg,#Const โดยสามารถแบ่งขอบเขตความหมายของชุดคำสั่งได้ ดังรูปที่ 5.5



รูปที่ 5.5 รูปแบบของชุดคำสั่งที่เป็นการดำเนินการระหว่างรีจิสเตอร์กับค่าคงที่

Opcode จะเป็นส่วนระบุถึงการทำงานดังรูปที่ 5.6

Opcode	Instruction	Opcode	Instruction
000	Sub Reg,#Const.	100	And Reg,#Const.
001	Add Reg,#Const.	101	Or Reg,#Const.
010	Suc Reg,#Const.	110	Xor Reg,#Const.
011	Adc Reg,#Const.	111	Mov Reg,#Const.

รูปที่ 5.6 ตาราง Opcode ของชุดคำสั่งที่มีการดำเนินการระหว่างรีจิสเตอร์กับค่าคงที่

Reg	Register
00	A
01	B
10	DPL
11	DPH

รูปที่ 5.7 ตารางการอ้างค่ารีจิสเตอร์ในชุดคำสั่ง

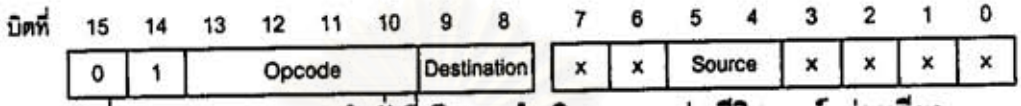
Reg ใช้ระบุตัวรีจิสเตอร์ โดยรีจิสเตอร์ A, B, DPL และ DPH จะถูกอ้างอิงด้วยค่าดังแสดงในตารางรูปที่ 5.7 ส่วน Constant Value เป็นค่าคงที่ขนาด 8 บิต

2. การดำเนินการระหว่างรีจิสเตอร์อย่างเดียว การดำเนินการแบบนี้แบ่งเป็น 2 ประเภท คือ

2.1 การดำเนินการระหว่างรีจิสเตอร์กับรีจิสเตอร์ คำสั่งที่อยู่ในประเภทนี้ ได้แก่ Sub, Add, Suc, Adc, And, Or, Xor และ Mov มีลักษณะการใช้งาน คือ Inst. Reg,Reg โดยที่ผลลัพธ์จะเก็บไว้ที่ รีจิสเตอร์ปลายทาง

2.2 การดำเนินการโดยรีจิสเตอร์เพียงตัวเดียว คำสั่งที่อยู่ในประเภทนี้ ได้แก่ Rol, Ror, Rlc และ Rrc มีลักษณะการใช้งาน คือ Inst. Reg

ซึ่งสามารถแบ่งขอบเขตความหมายของชุดคำสั่งได้ ดังรูปที่ 5.8



รูปที่ 5.8 รูปแบบของชุดคำสั่งที่เป็นการดำเนินการระหว่างรีจิสเตอร์อย่างเดียว

Opcode จะเป็นส่วนระบุการทำงานดังรูปที่ 5.9

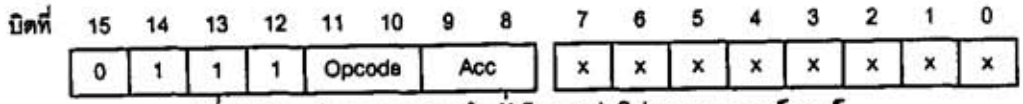
Opcode	Instruction	Opcode	Instruction
0000	Sub Dest,Source	0110	Xor Dest,Source
0001	Add Dest,Source	0111	Mov Dest,Source
0010	Suc Dest,Source	1000	Rol Destination
0011	Adc Dest,Source	1001	Ror Destination
0100	And Dest,Source	1010	Rlc Destination
0101	Or Dest,Source	1011	Rrc Destination

รูปที่ 5.9 ตาราง Opcode ของชุดคำสั่งที่มีการดำเนินการระหว่างรีจิสเตอร์อย่างเดียว

- Destination จะเป็นส่วนที่ใช้ระบุถึงรีจิสเตอร์ปลายทาง ซึ่งผลลัพธ์จะเก็บที่รีจิสเตอร์นี้ และทำหน้าที่เป็นรีจิสเตอร์ตัวตั้ง โดยรีจิสเตอร์ A, B, DPL และ DPH จะถูกอ้างอิงด้วยค่าตามตารางรูปที่ 5.7 ในกรณีที่เป็นการดำเนินการของรีจิสเตอร์ตัวเดียว ส่วนนี้จะใช้เป็นตัวระบุถึง รีจิสเตอร์ต้นทางด้วย

- Source จะเป็นส่วนระบุตัวรีจิสเตอร์ต้นทาง โดยรีจิสเตอร์ A, B, DPL และ DPH จะถูกอ้างอิงตามตารางรูปที่ 5.7 (ไม่ใช้ในกรณีที่เป็นการดำเนินการของรีจิสเตอร์ตัวเดียว)

3. *การโหลดค่าโปรแกรมเคาท์เตอร์* คำสั่งในกลุ่มนี้มีเพียง 2 คำสั่ง คือ Ld_pcl และ Ld_pch มีลักษณะการใช้งาน คือ Inst. Acc โดยสามารถแบ่งขอบเขตความหมายของชุดคำสั่งได้ ดังรูปที่ 5.10



รูปที่ 5.10 รูปแบบของชุดคำสั่งโหลดค่าโปรแกรมเคาท์เตอร์

Opcode	Instruction
00	Ld_pcl Acc
01	Ld_pch Acc

รูปที่ 5.11 ตาราง Opcode ของชุดคำสั่งโหลดค่าโปรแกรมเคาท์เตอร์

Acc	Register
00	A
01	B
10	DPL
11	DPH

รูปที่ 5.12 ตารางการอ้างค่ารีจิสเตอร์ในชุดคำสั่ง

Acc เป็นส่วนที่ใช้ระบุรีจิสเตอร์ปลายทางที่จะโหลดค่าของโปรแกรมเคาท์เตอร์เข้าไป ซึ่งสามารถโหลดค่าเข้าไปที่รีจิสเตอร์ A หรือ B เท่านั้น โดยใช้ค่าดังตารางรูปที่ 5.12 ในการอ้างถึง

4. การ Load และ Store ชุดคำสั่งนี้สามารถแบ่งขอบเขตของชุดคำสั่งได้ดังรูปที่ 5.13 ซึ่งแบ่งประเภทของการ Load และ Store ตามการอ้างอิงได้ 4 ประเภทของคำสั่ง คือ

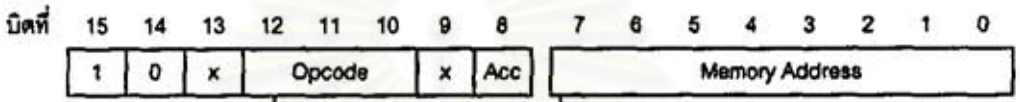
4.1 การอ้างอิงถึงหน่วยความจำทางอ้อม (Indirect Access Memory) เป็นชุดคำสั่งที่อ้างอิงถึงหน่วยความจำ โดยใช้ตัวชี้ข้อมูล (Data Pointer) คือ รีจิสเตอร์ DPH และ DPL ชุดคำสั่งประเภทนี้มีลักษณะการใช้งาน คือ Inst. Acc

4.2 การอ้างอิงถึงพอร์ตเข้าออก (Mapped I/O) เป็นชุดคำสั่งที่อ้างอิงถึงพอร์ตเข้าออก โดยใช้ข้อมูลตำแหน่งที่อยู่ 8 บิต ที่มีค่าตั้งแต่ 00000000 ถึง 00001110 ชุดคำสั่งประเภทนี้มีลักษณะการใช้งาน คือ Inst. Acc,MAP หรือ Inst. MAP,Acc ขึ้นอยู่กับทิศทางการ Load หรือ Store ข้อมูล

4.3 การอ้างอิงถึงหน่วยความจำโดยตรง (Direct Access Memory) เป็นชุดคำสั่งที่อ้างอิงถึงหน่วยความจำโดยตรง ซึ่งสามารถอ้างอิงได้เพียงตั้งแต่ตำแหน่งที่ 8000h ถึง 80FFh

เท่านั้น ชุดคำสั่งประเภทนี้มีลักษณะการใช้งาน คือ Inst. Acc,@ หรือ Inst. @,Acc ขึ้นอยู่กับทิศทางการ Load หรือ Store ข้อมูล

4.4 การอ้างอิงถึงตัวแปลงสัญญาณเชิงอุปมานเป็นสัญญาณเชิงเลข เป็นชุดคำสั่งที่อ้างอิงถึงข้อมูลจากตัวแปลงสัญญาณเชิงอุปมานเป็นสัญญาณเชิงเลข (A/D) การอ้างอิงนี้มีเพียงคำสั่งเดียว คือ Ld_adc Acc



รูปที่ 5.13 รูปแบบของชุดคำสั่ง Load และ Store

Opcode จะเป็นส่วนระบุการทำงานดังรูปที่ 5.14

Opcode	Instruction
000	Lod Acc
001	Lod Acc,MAP
010	Lod Acc,@
011	Lod_adc Acc
100	Sto Acc
101	Sto MAP,Acc
110	Sto @,Acc

รูปที่ 5.14 ตาราง Opcode ของชุดคำสั่ง Load และ Store

Acc จะเป็นส่วนระบุตัวรีจิสเตอร์ที่ใช้ Load หรือ Store ซึ่งอ้างอิงรีจิสเตอร์ตามตารางรูปที่ 5.12 และ Memory Address ในกรณีที่เป็นการอ้างอิงถึงหน่วยความจำโดยตรง จะเป็นค่าตำแหน่งที่อยู่ (Address) ไบต์ล่างของหน่วยความจำที่ใช้เพื่อ Load หรือ Store และใช้เป็นตำแหน่งของ Mapped I/O ที่อ้างอิงตำแหน่งที่อยู่ตามตารางรูปที่ 5.15 และ 5.16

Mapped I/O Address	Store to	Bit Number							
		7	6	5	4	3	2	1	0
00000000	CTRL	CapClr	x	x	x	x	x	x	x
00000001	P1	Port 1 (8-bit)							
00000010	P2	X	x	x	x	Port 2 (4-bit)			
00000011	RM_CT	RmClr	x	x	x	x	x	x	x
00000100	ROW0	X	x	x	x	ROW0 OSD data			
00000101	ROW1	X	x	x	x	ROW1 OSD data			
00000110	ROW2	x	x	x	x	ROW2 OSD data			
00000111	ROW3	x	x	x	x	ROW3 OSD data			
00001000	OSD_CT	x	x	x	x	Thai	Page	Scroll	x2
00001001	PWM1	x	x	PWM1 (6-bit) data					
00001010	PWM2	x	x	PWM2 (6-bit) data					
00001011	PWM3	x	x	PWM3 (6-bit) data					
00001100	PWM4	x	x	PWM4 (6-bit) data					
00001101	PWM0L	x	PWM0L (7-bit) low data						
00001110	PWM0H	x	PWM0H (7-bit) high data						

รูปที่ 5.15 ตารางตำแหน่งที่อยู่ของ Mapped I/O ที่ใช้กับคำสั่ง Store

Mapped I/O Address	Load from	Bit Number							
		7	6	5	4	3	2	1	0
00000000	P0	x	x	x	x	x	Port 0 (3-bit)		
00000001	CHAR1	Caption Data with Parity Checked (First Byte)							
00000010	CHAR2	Caption Data with Parity Checked (Second Byte)							
00000011	RM_DAT	x	x	Remote Control Data					
00000100	CAP_FLAG	x	x	x	x	x	x	x	Flag

รูปที่ 5.16 ตารางตำแหน่งที่อยู่ของ Mapped I/O ที่ใช้กับคำสั่ง Load

- CTRL เป็นรีจิสเตอร์สำหรับตั้งค่าใหม่ของตัวประมวลผลขั้นต้นสำหรับคำบรรยายภาพแบบซ่อนได้ หากบิตที่ 7 (CapClr) มีค่าเป็น '1'

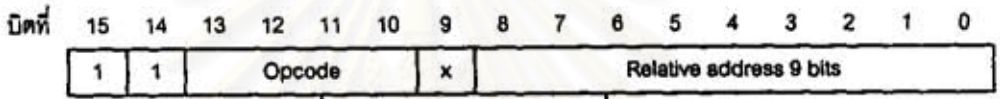
- CHAR1 และ CHAR2 เป็นรีจิสเตอร์ขนาด 8 บิต ที่เก็บข้อมูลคำบรรยายภาพแบบซ่อนได้ที่ได้รับการตรวจสอบภาวะเสมอมูล (Parity Check) โดยในบิตที่ 7 หากมีค่าเป็น '1' หมายถึงการตรวจสอบภาวะเสมอมูลคี่ (Odd Parity) ผ่าน แต่ถ้ามีค่าเป็น '0' หมายถึงไม่ผ่านการตรวจภาวะเสมอมูลคี่

- CAP_FLAG เป็นรีจิสเตอร์แสดงสถานะของข้อมูลคำบรรยายภาพแบบซ่อนได้ ซึ่งหากค่าของ FLAG เป็น '1' หมายถึงมีข้อมูลคำบรรยายภาพใหม่เข้ามา แต่ถ้าหากมีค่าเป็น '0' หมายถึงข้อมูลใหม่ยังไม่เข้ามา หลังจากที่มีการตั้งค่าใหม่ที่รีจิสเตอร์ CTRL
- P0 (Port 0) เป็นรีจิสเตอร์รับข้อมูลขาเข้าขนาด 3 บิต จากพอร์ตที่ 0
- P1 (Port 1) เป็นรีจิสเตอร์ส่งข้อมูลขาออกขนาด 8 บิต ไปที่พอร์ตที่ 1
- P2 (Port 2) เป็นรีจิสเตอร์ส่งข้อมูลขาออกขนาด 4 บิต ไปที่พอร์ตที่ 2
- RM_CT เป็นรีจิสเตอร์สำหรับตั้งค่าใหม่ของตัวประมวลผลขั้นต้นสำหรับเครื่องควบคุมระยะไกล หากบิตที่ 7 (RmClr) มีค่าเป็น '1'
- RM_DAT เป็นรีจิสเตอร์ข้อมูลรหัสขนาด 6 บิต จากตัวประมวลผลขั้นต้นสำหรับเครื่องควบคุมระยะไกล
- ROW0-ROW3 เป็นรีจิสเตอร์ควบคุมตำแหน่งการแสดงบรรทัดบนหน้าจอของตัวแสดงผลบนหน้าจอ แต่ละรีจิสเตอร์มีขนาด 4 บิต หมายถึงตำแหน่งที่จะแสดงในบรรทัดที่ 1 ถึง 15 โดยคำนวณค่าจาก ((ตำแหน่งบรรทัด + 2)) MOD 16)
- OSD_CT เป็นรีจิสเตอร์ควบคุมตัวแสดงผลบนหน้าจอ ซึ่งประกอบด้วย
    - บิต Thai และ Page จะเป็นตัวบอกตำแหน่งของหน่วยความจำแสดงผล (Display Memory) โดยบิต Thai หากเป็น '1' จากอ้างอิงถึงตำแหน่งของช่องสัญญาณ 2 (CC2) แต่ถ้าเป็น '0' จะอ้างอิงถึงตำแหน่งของช่องสัญญาณ 1 (CC1) และบิต Page จะอ้างอิงถึงหน้าแสดงผลที่ 0 และ 1
    - บิต Scroll ใช้ในการควบคุมตัวแสดงผลบนหน้าจอให้แสดงการเลื่อนบรรทัดขึ้น
    - บิต x2 ใช้ควบคุมขนาดของตัวอักษรที่แสดงตัวอักษรขนาดกว้าง 2 เท่า และสูง 2 เท่า หากบิตนี้มีค่าเป็น '1' และแสดงผลเป็นตัวอักษรขนาดปกติหากบิตนี้มีค่าเป็น '0'
- PWM1-PWM4 เป็นรีจิสเตอร์ขนาด 7 บิต ใช้ควบคุมความกว้างของสัญญาณมอดูเลตความกว้างพัลส์ที่มีความละเอียด 7 บิต (รีจิสเตอร์ PWMx ประกอบด้วยข้อมูล 6 บิต กับ '0' ที่บิต

ต่ำสุด ทำให้รีจิสเตอร์ที่ใช้เก็บข้อมูลมีขนาดเล็กลง เพื่อลดความหนาแน่นของวงจรภายใน ซึ่งมีรายละเอียดในหัวข้อที่ 7.2 ปัญหาในการทำงาน)

- PWM0L และ PWM0H เป็นรีจิสเตอร์ขนาด 7 บิต ที่แบ่งเป็นไบต์ล่างและไบต์บนของ รีจิสเตอร์ควบคุมความกว้างของสัญญาณมอดูเลตความกว้างพัลส์ที่มีความละเอียด 14 บิต

5. การกระโดด ชุดคำสั่งสำหรับการกระโดดแบ่งเป็น 2 ประเภทคือ การกระโดดแบบสัมพัทธ์ และการกระโดดด้วย Index ซึ่งมีเพียงคำสั่งเดียว คือ คำสั่ง Jmp โดยแบ่งขอบเขตแต่ละบิตของชุดคำสั่งแสดงดังรูปที่ 5.16



รูปที่ 5.17 รูปแบบของชุดคำสั่งกระโดด

Opcode จะเป็นส่วนระบุการทำงานดังรูปที่ 5.17

Opcode	Instruction	Opcode	Instruction
0000	Jnc Rel 9 bits	0011	Jz Rel 9 bits
0001	Jc Rel 9 bits	01xx	Jr Rel 9 bits
0010	Jnz Rel 9 bits	1xxx	Jmp

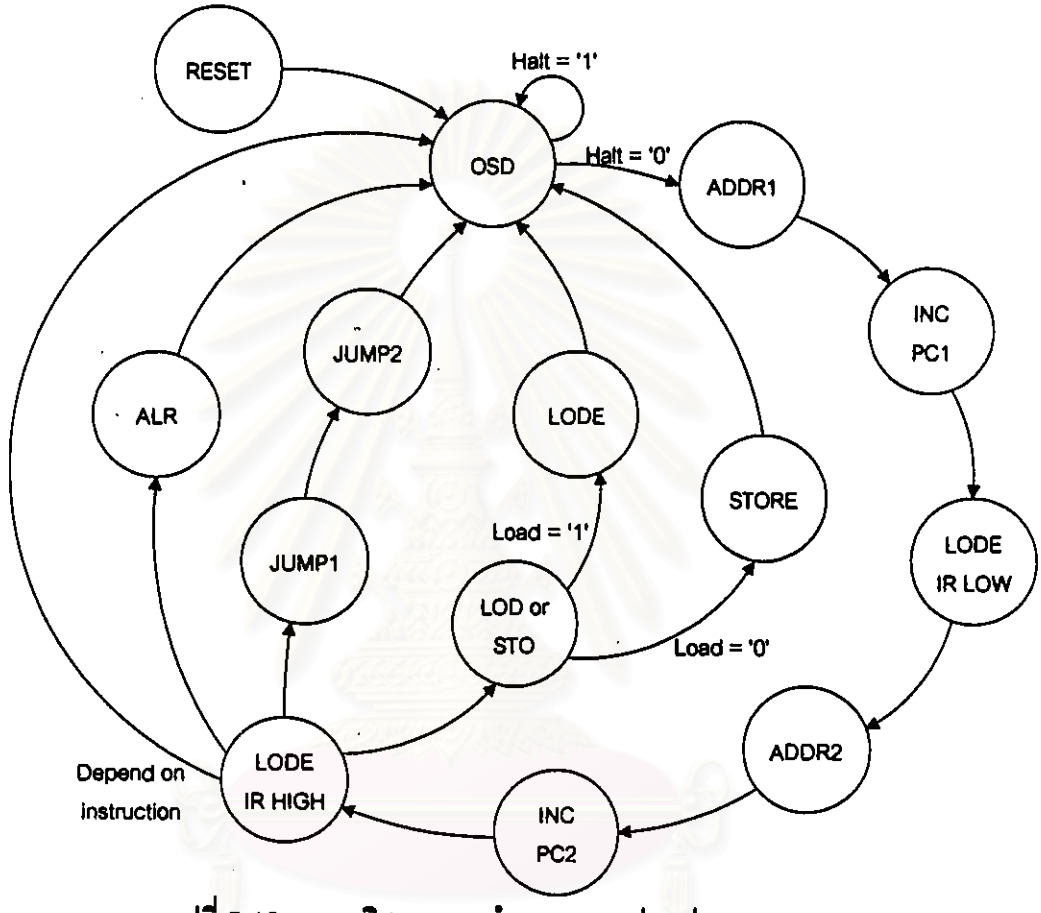
รูปที่ 5.18 ตาราง Opcode ของชุดคำสั่งกระโดด

Relative address 9 bits เป็นส่วนกำหนดค่าสัมพัทธ์ในการกระโดด มีขนาด 9 บิต โดยบิตที่ 8 จะเป็นบิตที่บอกทิศทางการกระโดด ถ้าบิตที่ 8 เป็น '0' หมายถึงกระโดดไปในทิศทางที่ลดค่าของโปรแกรมเคาท์เตอร์ แต่ถ้าเป็น '1' หมายถึงกระโดดไปในทิศทางที่เพิ่มค่าของโปรแกรมเคาท์เตอร์

5.4 จังหวะการทำงานของหน่วยประมวลผลกลาง

หน่วยประมวลผลกลางทำงานที่ความถี่ครึ่งหนึ่งของสัญญาณนาฬิกาที่ตัวแสดงผลบนหน้าจอใช้งาน คือ 6 เมกะเฮิรตซ์ โดยแต่ละขั้นตอนการทำงานจะใช้ระยะเวลา 1 คาบสัญญาณ

นาฬิกา ซึ่งสามารถอธิบายขั้นตอนการทำงานแต่ละขั้นได้ด้วยแผนภูมิการเปลี่ยนสเตท ดังรูปที่ 5.18 แต่ละสเตทมีการทำงานดังต่อไปนี้



รูปที่ 5.19 แผนภูมิสเตทการทำงานของหน่วยประมวลผลกลาง

1. Reset เป็นสเตทเริ่มต้นการทำงานของหน่วยประมวลผลกลาง ในสเตทนี้จะเป็นการตั้งค่าให้เริ่มต้นให้กับรีจิสเตอร์ทุกตัวมีค่าเป็นศูนย์ สเตทนี้จะเกิดขึ้นหลังการจ่ายไฟให้กับชิป หรือมีการกดปุ่ม Reset

2. OSD เป็นสเตทที่หน่วยประมวลผลกลางจะหยุดทำงานชั่วขณะ และยอมให้ตัวแสดงผลบนหน้าจอใช้งานทั้งบัสข้อมูล (Data Bus) และบัสตำแหน่ง (Address Bus) โดยจะคงอยู่ที่สเตทนี้หากสัญญาณ Halt จากตัวแสดงผลบนหน้าจอมีค่าเป็น '1' และจะเปลี่ยนเป็นสเตทถัดไป คือ สเตท ADDR1 หากสัญญาณ Halt มีค่าเป็น '0'

3. *ADDR1* เป็นสเตทที่หน่วยประมวลผลกลางส่งค่าตำแหน่งที่อยู่ของชุดคำสั่งไบต์ต่ำจากโปรแกมเคาน์เตอร์ ไปยังบัสตำแหน่งเพื่อเตรียมการอ่านค่าชุดคำสั่งไบต์ต่ำมา และในสเตทนี้จะเพิ่มค่าของโปรแกรมเคาน์เตอร์ที่ไบต์ต่ำด้วยหน่วยคำนวณ, ตรรก และหมุน

4. *INC PC1* เป็นสเตทที่จะเก็บค่าของโปรแกรมเคาน์เตอร์ไบต์ต่ำ และเพิ่มค่าไบต์สูงของโปรแกรมเคาน์เตอร์หากมีการทดค่าจากไบต์ต่ำ

5. *LOAD IR LOW* เป็นสเตทที่หน่วยประมวลผล กลางจะอ่านค่าไบต์ต่ำของชุดคำสั่งเข้ามายัง รีจิสเตอร์คำสั่ง และเก็บค่าของโปรแกรมเคาน์เตอร์ไบต์สูงหากมีการทดค่าจากไบต์ต่ำ

6. *ADDR2* เป็นสเตทที่หน่วยประมวลผลกลางส่งค่าตำแหน่งที่อยู่ของชุดคำสั่งไบต์สูงจากโปรแกมเคาน์เตอร์ ไปยังบัสตำแหน่งเพื่อเตรียมการอ่านค่าชุดคำสั่งไบต์สูงมา และในสเตทนี้จะเพิ่มค่าของโปรแกรมเคาน์เตอร์ที่ไบต์ต่ำโดยใช้หน่วยคำนวณ, ตรรก และหมุน

7. *INC PC2* เป็นสเตทที่จะเก็บค่าของโปรแกรมเคาน์เตอร์ไบต์ต่ำ และเพิ่มค่าไบต์สูงของโปรแกรมเคาน์เตอร์หากมีการทดค่าจากไบต์ต่ำ

8. *LOAD IR HIGH* เป็นสเตทที่หน่วยประมวลผลกลางจะอ่านค่าไบต์สูงของชุดคำสั่งเข้ามายังรีจิสเตอร์คำสั่ง และในสเตทนี้เป็นสเตทที่จะทำงานตามประเภทของชุดคำสั่งที่ได้รับ คือ

- ชุดคำสั่งประเภทคำนวณ, ตรรก และหมุน ส่วนถอดรหัสและควบคุมภายในหน่วยประมวลผลกลางจะกำหนดค่าให้กับรีจิสเตอร์ TMP1 กับ TMP2 และกำหนดการทำงานของหน่วยคำนวณ, ตรรก และหมุน ตามชุดคำสั่งที่ได้รับมา อีกทั้งเก็บค่าโปรแกรมเคาน์เตอร์ไบต์สูงที่ได้รับการเพิ่มค่า ก่อนที่จะไปยังสเตท ALR ต่อไป
- ชุดคำสั่ง Load หรือ Store จะทำการเก็บค่าโปรแกรมเคาน์เตอร์ไบต์สูงที่ได้รับการเพิ่มค่า แล้วไปยังสเตท Load or Store ต่อไป
- ชุดคำสั่งประเภทกระโดด หากเป็นชุดคำสั่งที่เป็นการกระโดดแบบสัมพัทธ์ ส่วนถอดรหัสและควบคุมภายในหน่วยประมวลผลกลางจะกำหนดค่าให้กับรีจิสเตอร์ TMP1 กับ TMP2 เพื่อทำการคำนวณค่าไบต์ต่ำของโปรแกรมเคาน์เตอร์ตามแต่การกระโดดสัมพัทธ์นั้นๆ และไปยังสเตทต่อไป คือ สเตท Jump1 แต่ถ้าหากว่าเป็นคำสั่งกระโดดด้วย Index คือ คำสั่ง JMP จะเป็นการกำหนดค่าให้กับโปรแกรมเคาน์เตอร์ด้วยรีจิสเตอร์ DPH กับ DPL และไปยังสเตท OSD เป็นสเตทต่อไป

9. ALR เป็นสเตทที่ส่วนถอดรหัส และควบคุมภายในหน่วยประมวลผลกลางนำผลลัพธ์ที่ได้จากการดำเนินการโดยหน่วยคำนวณ, ตรรก และหมุน ในรีจิสเตอร์ R ไปเก็บยังรีจิสเตอร์ปลายทางตามคำสั่งที่ได้รับมา

10. LOAD or STORE เป็นสเตทที่เตรียมพร้อมในการอ่านหรือเขียนข้อมูล โดยจะส่งสัญญาณที่ใช้ในการควบคุมการอ่าน หรือเขียนออกไป ก่อนที่จะไปยังสเตท LOAD หรือ STORE ต่อไป ตามแต่คำสั่งที่ได้รับ

11. LOAD เป็นสเตทที่อ่านข้อมูลจากหน่วยความจำ หรือ Mapped I/O เข้ามายังรีจิสเตอร์ จากนั้นจะไปยังสเตท OSD ต่อไป

12. STORE เป็นสเตทที่จะเขียนข้อมูลไปยังหน่วยความจำ หรือ Mapped I/O จากรีจิสเตอร์ จากนั้นจะไปยังสเตท OSD ต่อไป

13. JUMP1 เป็นสเตทที่จะเก็บค่าของโปรแกรมเคาน์เตอร์ไบต์ต่ำที่ได้จากการคำนวณค่าโดยหน่วยคำนวณ, ตรรก และหมุน จากนั้นไปยังสเตท JUMP2

14. JUMP2 เป็นสเตทที่จะเก็บค่าโปรแกรมเคาน์เตอร์ไบต์สูงที่ได้จากการคำนวณค่าโดยหน่วยคำนวณ, ตรรก และหมุน จากนั้นไปยังสเตท OSD