บทที่ 3
การออกแบบวงจรหน่วยประมวลผล

หน่วยประมวลผลที่ออกแบบในงานวิจัยนี้ ได้พัฒนามาจากหน่วยประมวลผลแบบแสตก [15] ซึ่งได้ออกแบบเพื่อให้ขนาดของวงจรเล็กเพื่อประหยัดทรัพยากร อีกทั้งมีชุดคำสั่งแบบแสตกช่วยให้โปรแกรมของหน่วยประมวลผลนี้มีขนาดเล็ก แต่ในขณะเดียวกันการทำงานแบบแสตกได้จำกัดสมรรถนะของหน่วยประมวลผล เนื่องจากการทำงานแบบแสตกมีการเข้าถึงหน่วยความจำ (Memory) บ่อยครั้ง ในหน่วยประมวลผลของงานวิจัยนี้จึงได้ตัดการทำงานที่ดำเนินการกับข้อมูลที่อยู่บนแสตกออกไป เพื่อเพิ่มสมรรถนะในการทำงานของหน่วยประมวลผล การออกแบบวงจรหน่วยประมวลผลสามารถแบ่งเป็นขั้นตอนย่อยๆได้ดังนี้

3.1 การออกแบบชุดคำสั่ง

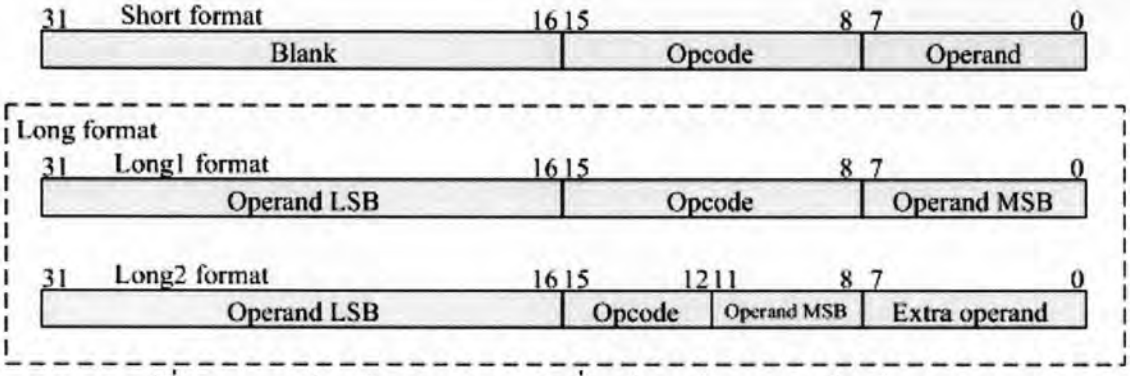
หน่วยประมวลผลของงานวิจัยนี้ ใช้ชุดคำสั่งที่พัฒนามาจากชุดคำสั่งของหน่วยประมวลผลแบบแสตก [15] โดยได้มีการปรับเปลี่ยนคำสั่งที่ไม่มีตัวถูกดำเนินการ (Operand) ในชุดคำสั่งเดิมให้เป็นคำสั่งแบบที่มีตัวถูกดำเนินการ 1 ตัว การทำเช่นนี้เพื่อให้หน่วยประมวลผลไม่มีการทำงานแบบแสตก ช่วยลดการเข้าถึงหน่วยความจำและเพิ่มประสิทธิภาพการทำงานได้ ชุดคำสั่งของหน่วยประมวลผลของงานวิจัยนี้สามารถแบ่งหมวดหมู่ได้เป็น 4 ประเภทดังแสดงในตารางที่ 3.1

ตารางที่ 3.1 คำสั่งในชุดคำสั่งของหน่วยประมวลผลของงานวิจัยนี้

ประเภท	คำสั่ง
เลขคณิตและตรรกะ - Binary operation with local variable type operand - Binary operation with immediate type operand - Unary operation	 ADD, SUB, MUL*, DIV*, OR, AND, XOR, MOD*, ADDC, SUBC, EQ, NEQ, GT, GE, LT, LE ADDI, SUBI, MULI*, DIVI*, ORI, ANDI, XORI, MODI*, ADDCI, SUBCI, EQI, NEQI, GTI, GEI, LTI, LEI NOT, SHL0, SHR0, SHL, SHR, SHLC, SHRC, SHL4, SHR4
จัดการข้อมูล	LD, LDL, ST, STL, LDX, STV, GET, PUT, INC, DEC
ควบคุม	JMP, JMPL, JT, JTL, JF, JFL, JC, JCL, JNC, JNCL, IJX, CALL, RET
พิเศษ	LIT, LITL, NOP, BLANK

* คำสั่งดังกล่าวยังไม่สามารถใช้งานได้

ชุดคำสั่งของหน่วยประมวลผลนี้มี 3 รูปแบบ คือ รูปแบบสั้น (Short format) รูปแบบยาว (Long format) มี 2 แบบคือ รูปแบบยาว 1 (Long1 format) และรูปแบบยาว 2 (Long2 format) โดยสำหรับชุดคำสั่งสั้นมีขนาด 16 บิต ประกอบด้วยรหัสดำเนินการ (Opcode) 8 บิต และตัวถูกดำเนินการ (Operand) 8 บิต ชุดคำสั่งยาวทั้งสองแบบมีขนาด 32 บิต โดยรูปแบบยาว 1 ประกอบด้วยรหัสดำเนินการ 8 บิตและตัวถูกดำเนินการ 24 บิต และรูปแบบยาว 2 ประกอบด้วยรหัสดำเนินการ 4 บิต ตัวถูกดำเนินการ 20 บิตและตัวถูกดำเนินการเพิ่มเติม 8 บิต ดังแสดงในรูปที่ 3.1



รูปที่ 3.1 แสดงส่วนประกอบของชุดคำสั่งรูปแบบต่างๆของหน่วยประมวลผล

ในบางคำสั่งที่ต้องการใช้ตัวถูกดำเนินการที่มีค่ามากๆ ได้แก่ LD, ST, JMP, JT, JF, JC และ JNC ขนาดของตัวถูกดำเนินการของคำสั่งรูปแบบเล็กนั้นมีขนาดเล็กเกินไป ไม่เอื้ออำนวยต่อการพัฒนาโปรแกรมของหน่วยประมวลผล จึงได้มีการเพิ่มคำสั่ง LDL, STL, JMPL, JTL, JFL, JCL และ JNCL ซึ่งเป็นคำสั่งรูปแบบยาว เพื่อเพิ่มขนาดของตัวถูกดำเนินการที่อยู่ในคำสั่ง โดยการทำงานของคำสั่งทั้ง 2 รูปแบบนั้นเหมือนกันทุกประการ ตารางที่ 3.2 แสดงรูปแบบที่มีในคำสั่งแต่ละประเภทของหน่วยประมวลผลนี้

ตารางที่ 3.2 รูปแบบของชุดคำสั่งแต่ละประเภทของหน่วยประมวลผล

คำสั่ง	คำสั่งสั้น	คำสั่งยาว 1	คำสั่งยาว 2
ประเภทเลขคณิตและตรรกะทุกคำสั่ง	●		
ประเภทจัดการข้อมูล	●	●	
- ทุกคำสั่งยกเว้น LD, LDL, ST, STL	●		
- LD, LDL, ST, STL	●	●	
ประเภทควบคุม	●	●	●
- RET	●		
- JMP, JMPL, All conditional jump	●	●	
- CALL, IJX			●
พิเศษ (คำสั่งLIT, LITL)	●	●	

คำสั่งประเภทเลขคณิตและตรรกะมีการกำหนดเลขที่อยู่ (Addressing mode) ของตัวถูกดำเนินการ (Operand) 2 แบบ คือ ตัวดำเนินการแบบตัวแปรเฉพาะที่ (Local variable) และตัวดำเนินการแบบทันที (Immediate)

3.2 การออกแบบทางเดินข้อมูล

ทางเดินข้อมูล (Data path) ของหน่วยประมวลผลของงานวิจัยนี้ พัฒนาต่อมาจากทางเดินข้อมูลของหน่วยประมวลผลแบบแสตก จากการปรับปรุงชุดคำสั่งเพื่อกำจัดการทำงานแบบแสตก ทางเดินข้อมูลของหน่วยประมวลผลนี้จึงไม่มีรีจิสเตอร์ (Register) ที่ทำงานเกี่ยวข้องกับแสตกเช่น รีจิสเตอร์แสตกพอยเตอร์ (SP: Stack pointer) และในหน่วยประมวลผลนี้ได้ขยายขนาดของทางเดินข้อมูลจากเดิม 16 บิตเป็น 32 บิต

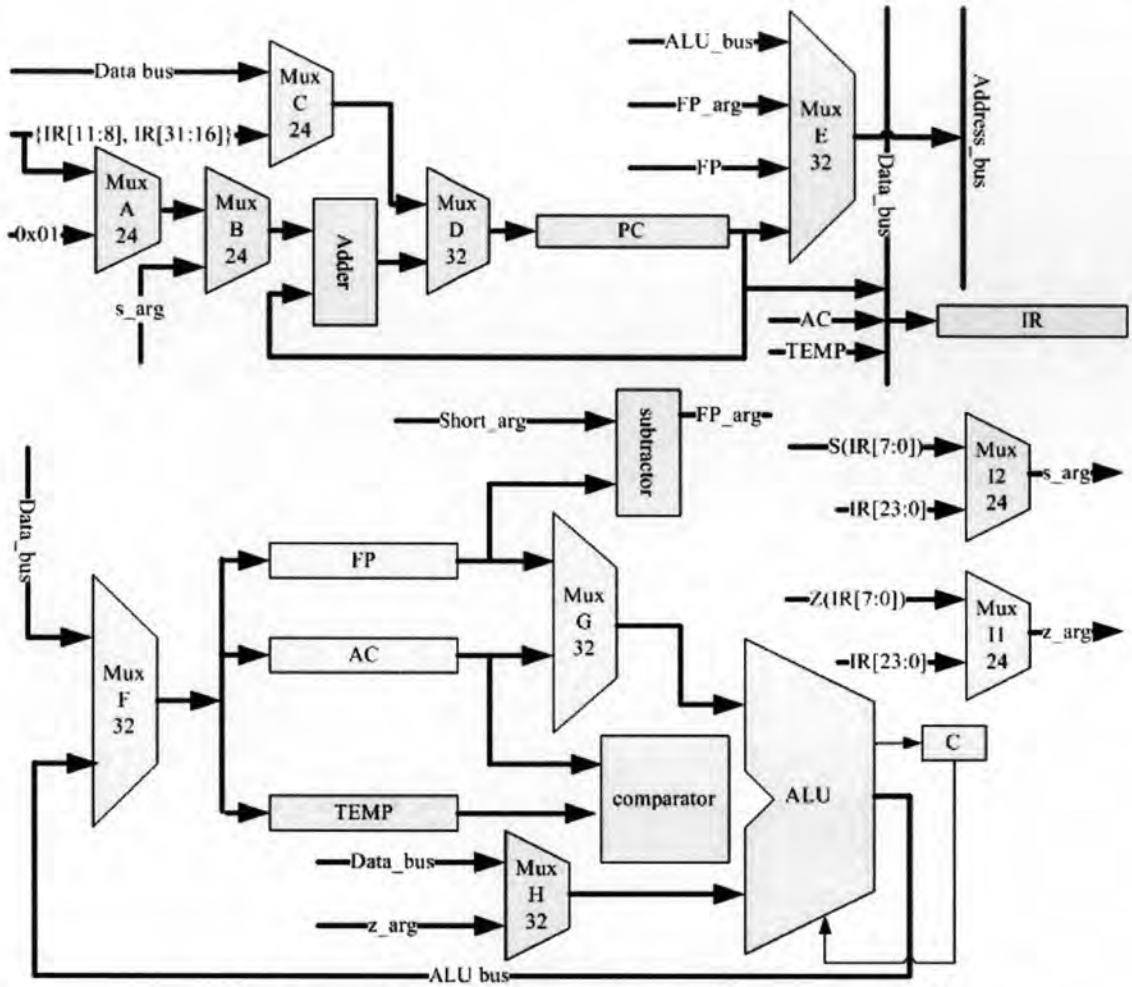
ทางเดินข้อมูลของหน่วยประมวลผลของงานวิจัยนี้ออกแบบให้วงจรมีขนาดเล็ก ดังแสดงในรูปที่ 3.2 ทางเดินข้อมูลของหน่วยประมวลผลนี้ประกอบด้วยรีจิสเตอร์ 6 ตัวซึ่งเป็นรีจิสเตอร์ขนาด 32 บิตจำนวน 5 ตัวและรีจิสเตอร์ขนาด 1 บิตอีก 1 ตัว คือ

1. AC (Accumulator) สำหรับเก็บค่าผลลัพธ์ที่ได้จากการประมวลผล
2. FP (Frame pointer) ทำหน้าที่เป็นตัวชี้ตำแหน่งของแอคทิเวชันเรคคอร์ด (Activation record) ที่กำลังทำงานอยู่
3. PC (Program counter) ทำหน้าที่เป็นตัวชี้ตำแหน่งของคำสั่งถัดไป
4. IR (Instruction register) รีจิสเตอร์เก็บคำสั่งที่หน่วยประมวลผลกำลังทำงานอยู่
5. TEMP ทำหน้าที่เก็บค่าเพิ่มเติมในบางคำสั่ง
6. C รีจิสเตอร์ ขนาด 1 บิตทำหน้าที่เก็บตัวทด (Carry) ที่ได้จากการคำนวณของวงจรหน่วยคำนวณและตรรกะ (ALU)

รีจิสเตอร์ทั้ง 6 ตัวที่ได้กล่าวไปข้างต้น มีเพียงรีจิสเตอร์ AC เท่านั้นที่อนุญาตให้ผู้พัฒนาโปรแกรมตั้งค่าได้ การตั้งค่าของรีจิสเตอร์ AC ทำได้โดยใช้คำสั่ง LIT และ LITL รีจิสเตอร์อื่นๆนั้นไม่สามารถทำการเปลี่ยนค่าโดยตรงได้

รีจิสเตอร์ FP ซึ่งชี้ตำแหน่งของแอคทิเวชันเรคคอร์ด (Activation record) จะเปลี่ยนแปลงด้วยคำสั่ง CALL และ RET ซึ่งค่าที่เปลี่ยนนี้จะต้องสอดคล้องกับขนาดของแอคทิเวชันเรคคอร์ด (Activation record) ของโปรแกรมย่อยที่ถูกเรียก รายละเอียดของแอคทิเวชันเรคคอร์ด จะกล่าวต่อไปในบทที่ 6 การพัฒนาโปรแกรม

รีจิสเตอร์ PC ทำหน้าที่เป็นตัวชี้ตำแหน่งของคำสั่งถัดไป ค่าของ PC จะเลื่อนไปที่ตำแหน่งคำสั่งถัดไปโดยอัตโนมัติเมื่อทำการดึงคำสั่งเสร็จสิ้น โดยค่านั้นเปลี่ยนแปลงได้เมื่อคำสั่งที่ทำงานเป็นคำสั่งประเภทควบคุม



รูปที่ 3.2 ทางเดินข้อมูลของหน่วยประมวลผลที่พัฒนาในงานวิจัยนี้

รีจิสเตอร์ IR ทำหน้าที่เก็บคำสั่งที่หน่วยประมวลผลกำลังทำงานอยู่ รีจิสเตอร์นี้ทำงานโดยอัตโนมัติไม่มีคำสั่งใดที่เปลี่ยนค่าในรีจิสเตอร์นี้ รีจิสเตอร์ TEMP เป็นรีจิสเตอร์เพิ่มเติมเพื่อทำหน้าที่เก็บค่าที่จำเป็นกับการประมวลผลในบางคำสั่ง และรีจิสเตอร์ C ซึ่งทำหน้าที่เก็บตัวทด (Carry) ที่ได้จากการคำนวณของวงจรหน่วยคำนวณและตรรกะ (ALU)

3.3 การกำหนดความหมายของคำสั่ง

ในส่วนนี้จะเป็นการกำหนดว่าเมื่อหน่วยประมวลผลทำงานคำสั่งหนึ่งๆเสร็จแล้ว จะมีการเปลี่ยนแปลงอย่างไรเกิดขึ้นกับข้อมูลในหน่วยประมวลผลนี้หรือในหน่วยความจำ ทั้งนี้เพื่อให้เกิดความกระจ่างว่าคำสั่งแต่ละคำสั่งของหน่วยประมวลผลนี้ทำหน้าที่อะไร มีการดำเนินการอย่างไรกับข้อมูล ซึ่งจำเป็นในการออกแบบพฤติกรรมของหน่วยประมวลผลนี้ในบทถัดไป

คำสั่งประเภทเลขคณิตและตรรกะคำสั่งเหล่านี้ทำหน้าที่ด้านการคำนวณ โดยข้อมูลที่เป็นตัวตั้งของการคำนวณ คือ ค่าที่เก็บในรีจิสเตอร์ AC สำหรับการคำนวณที่มีตัวถูกดำเนินการ (Operand) 2 ตัว ข้อมูลอีกตัวจะถูกกำหนดเลขที่อยู่ (Addressing) โดยตัวถูกดำเนินการในคำสั่ง การ

กำหนดเลขที่อยู่นั้นมี 2 แบบ คือ แบบตัวแปรเฉพาะที่ (Local variable) และแบบทันที (Immediate) ความหมายของคำสั่งประเภทเลขคณิตและตรรกะแสดงในภาคผนวก ก ตารางที่ ก.2

คำสั่งประเภทจัดการข้อมูลทำหน้าที่ด้านการจัดการการเคลื่อนย้ายข้อมูลระหว่างหน่วยประมวลผลและหน่วยความจำ คำสั่ง LD, LDL, LDX และ GET ทำหน้าที่ดึงข้อมูลจากหน่วยความจำเข้ามาในหน่วยประมวลผล คำสั่ง ST, STL, STV และ PUT ทำหน้าที่นำข้อมูลจากหน่วยประมวลผลไปเก็บในหน่วยความจำ คำสั่ง PASS ทำหน้าที่ส่งผ่านพารามิเตอร์ (Parameter) จากแอคทิเวชันเรคคอร์ด (Activation reccord) ปัจจุบันไปยังแอคทิเวชันเรคคอร์ดถัดไป รายละเอียดของขั้นตอนการส่งผ่านพารามิเตอร์จะกล่าวต่อไปในบทที่ 6

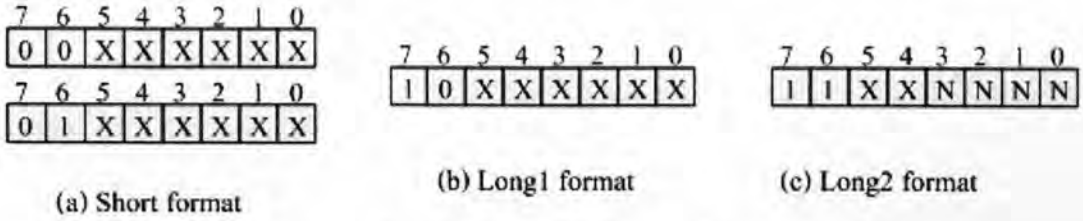
นอกจากนี้คำสั่ง INC และ DEC ในคำสั่งประเภทจัดการข้อมูลทำการเพิ่มและลดค่าของตัวแปรเฉพาะที่ (Local variable) ในแอคทิเวชันเรคคอร์ด (Activation reccord) ความหมายของคำสั่งประเภทจัดการข้อมูลแสดงในภาคผนวก ก ตารางที่ ก.4

คำสั่งประเภทควบคุม ทำหน้าที่ควบคุมการทำงานของหน่วยประมวลผล โดยประกอบด้วยคำสั่งกระโดด (Jump) ได้แก่ JMP และ JMPL ซึ่งเป็นการกระโดดแบบไม่มีเงื่อนไข คำสั่ง JT, JTL, JF, JFL, JC, JCL, JNC, JNCL และ IJX เป็นการกระโดดแบบมีเงื่อนไข คำสั่ง CALL และ RET ทำหน้าที่เรียกโปรแกรมย่อยและกลับคืนสู่โปรแกรมหลัก รวมทั้งสร้างแอคทิเวชั่นเรคคอร์ด (Activation reccord) ในตอนที่มีการเรียกโปรแกรมย่อย และทำลายแอคทิเวชันเรคคอร์ดใหม่เมื่อทำการกลับสู่โปรแกรมหลักด้วย ความหมายของคำสั่งประเภทควบคุม แสดงในภาคผนวก ก ตารางที่ ก.5

คำสั่งประเภทพิเศษ เช่น คำสั่ง LIT และ LITL ทำหน้าที่เขียนค่าเริ่มต้นที่ต้องการลงในรีจิสเตอร์ AC คำสั่ง NOP ไม่ทำให้เกิดการดำเนินการใดๆในหน่วยประมวลผลเป็นเวลา 1 รอบนาฬิกา (Cycle) ส่วนคำสั่ง BLANK นั้นเป็นคำสั่งพิเศษที่ใช้ในการอัดคำสั่งซึ่งจะกล่าวต่อไป คำสั่งนี้ไม่ทำให้เกิดการทำงานใดในหน่วยประมวลผล ความหมายของคำสั่งประเภทพิเศษ แสดงในภาคผนวก ก ตารางที่ ก.3

3.4 การแทนรหัสดำเนินการแต่ละคำสั่ง

การแทนรหัสดำเนินการ (Opcode) ให้แก่คำสั่งในชุดคำสั่งของหน่วยประมวลผลนี้ ชุดคำสั่งของหน่วยประมวลผลนี้มีรูปแบบ 3 รูปแบบดังที่ได้กล่าวไปแล้ว คือ รูปแบบสั้น (Short format) รูปแบบยาว 1 (Long1 format) และรูปแบบยาว 2 (Long2 format) เพื่อความง่ายต่อการแบ่งแยกรูปแบบของคำสั่ง บิตสูง (Most significant bit) จำนวน 2 บิตของรหัสดำเนินการจะถูกใช้ในการบอกรูปแบบของคำสั่ง ดังแสดงในรูปที่ 3.3



(a) Short format (b) Long1 format (c) Long2 format

รูปที่ 3.3 ส่วนของรหัสดำเนินการที่ใช้ในการแบ่งรูปแบบของคำสั่ง

จากรูปที่ 3.3 บิตที่ 6 และ 7 ของรหัสดำเนินการถูกใช้ในการระบุรูปแบบของคำสั่ง บิตดังกล่าวมีค่าเป็น 00 หรือ 01 เมื่อคำสั่งนั้นเป็นคำสั่งรูปแบบสั้น (Short format) เป็น 10 สำหรับคำสั่งแบบยาว 1 (Long1 format) และเป็น 11 สำหรับคำสั่งแบบยาว 2 (Long2 format) การที่กำหนดบิตเช่นนี้มีจุดประสงค์เพื่อให้การจำแนกรูปแบบของคำสั่งทำได้ง่ายและรวดเร็ว ซึ่งจะช่วยลดขนาดของวงจรหน่วยควบคุม (Control unit) ได้

จากการที่บิต 6 และ 7 ของรหัสดำเนินการ (Opcode) ถูกใช้ในการระบุรูปแบบของคำสั่ง จึงเหลือบิตอีก 6 บิตในการระบุคำสั่งที่อยู่ในรูปแบบสั้น (Short format) และรูปแบบยาว 1 (Long1 format) ส่วนคำสั่งในรูปแบบยาว 2 (Long2 format) มีบิตเหลือ 2 บิตสำหรับใช้ในการระบุคำสั่ง ค่ารหัสดำเนินการของคำสั่งในชุดคำสั่งของหน่วยประมวลผลนี้ แสดงในภาคผนวก ก ตารางที่ ก.6

3.5 การออกแบบสถานะและการออกแบบพฤติกรรมการทำงานของหน่วยประมวลผล

พฤติกรรมการทำงานระดับการขนถ่ายข้อมูลระหว่างรีจิสเตอร์ (Register transfer level: RTL) เป็นการกำหนดการเคลื่อนย้ายของข้อมูลในแต่ละสถานะการทำงานของหน่วยประมวลผล สถานะของหน่วยประมวลผลนี้ สามารถแบ่งออกได้เป็น 3 กลุ่ม ตามลำดับขั้นตอนการทำงานของหน่วยประมวลผลประกอบด้วย กลุ่มสถานะที่อยู่ในขั้นตอนการดึงคำสั่งและถอดรหัสคำสั่ง (Instruction fetch & decode phase) กลุ่มสถานะที่อยู่ในขั้นตอนการกระทำที่ 1 (Execute1 phase) และกลุ่มสถานะที่อยู่ในขั้นตอนการกระทำที่ 2 (Execute2 phase) ดังแสดงในรูปที่ 3.4 และ 3.5

หมายเหตุ การกล่าวว่าหน่วยประมวลผลทำงานอยู่ในขั้นตอนใดๆ นั้น ในที่นี้หมายความว่าหน่วยประมวลผลอยู่ในสถานะใดสถานะหนึ่ง ซึ่งสถานะดังกล่าวอยู่ในกลุ่มสถานะที่อยู่ในขั้นตอนการทำงานนั้นๆ เช่น เมื่อกล่าวว่าหน่วยประมวลผลทำงานในขั้นตอนการกระทำที่ 2 (Execute2 phase) หมายความว่า หน่วยประมวลผลกำลังอยู่ในสถานะหนึ่งที่อยู่ในขั้นตอนการกระทำที่ 2 ซึ่งได้แก่ ST_LDX2, ST_STV2, ST_INC2 และ ST_IJX2

ในทางกลับกัน เมื่อหน่วยประมวลผลอยู่ในสถานะหนึ่ง สถานะนั้นอยู่ในขั้นตอนใดก็สามารถกล่าวได้ว่า หน่วยประมวลผลทำงานอยู่ในขั้นตอนนั้นเช่นกัน

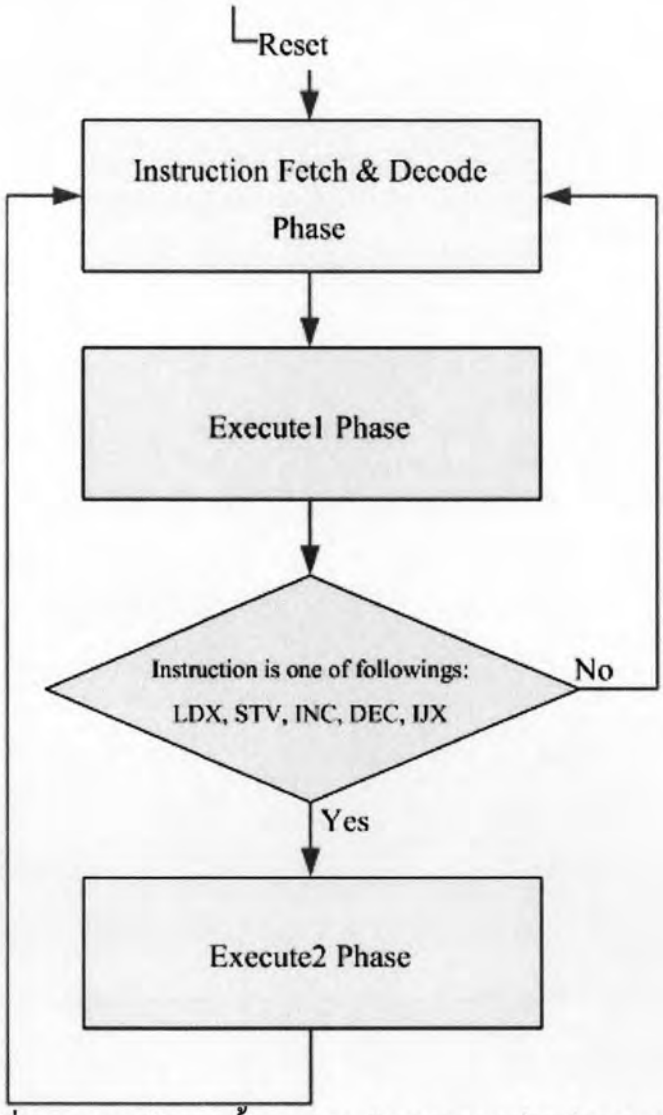
ขั้นตอนการทำงานของหน่วยประมวลผลนี้มีทั้งหมด 3 ขั้นตอนคือ ขั้นตอนการดึงคำสั่งและถอดรหัสคำสั่ง (Instruction fetch & decode phase) ขั้นตอนการกระทำที่ 1 (Execute1 phase)

และขั้นตอนการกระทำที่ 2 (Execute2 phase) ดังแสดงในรูปภาพที่ 3.4 โดยแต่ละขั้นตอนใช้เวลาในการทำงาน 1 รอบนาฬิกา (Clock cycle) การทำงานของหน่วยประมวลผลนี้จะเริ่มที่ขั้นตอนการดึงคำสั่งและถอดรหัสคำสั่ง จากนั้นจะเข้าสู่ขั้นตอนการกระทำที่ 1

สำหรับคำสั่งที่ไม่ใช่คำสั่ง LDX, STV, INC, DEC และ IJX การกระทำ (Execute) เสร็จสิ้นในขั้นตอนการกระทำที่ 1 หน่วยประมวลผลจะเข้าสู่ขั้นตอนการดึงคำสั่งและถอดรหัสคำสั่งอีกครั้งเพื่ออ่านคำสั่งถัดไปในโปรแกรม

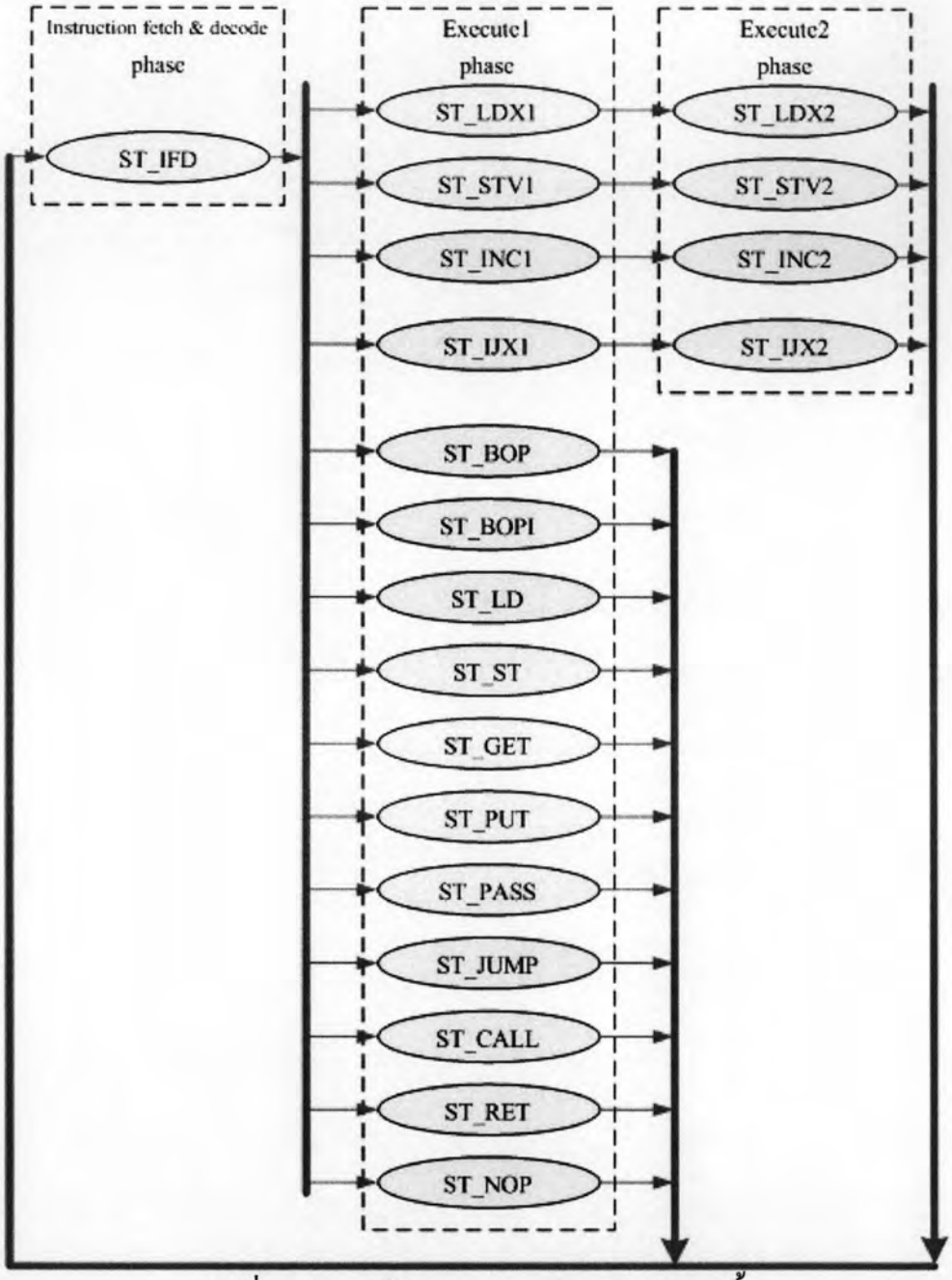
สำหรับคำสั่ง LDX, STV, INC, DEC และ IJX ซึ่งการกระทำ (Execute) ยังไม่เสร็จสิ้น หน่วยประมวลผลจะเข้าสู่ขั้นตอนการกระทำที่ 2 (Execute2 phase) เพื่อทำงานคำสั่งนั้นๆ ให้เสร็จ จากนั้นจึงเข้าสู่ขั้นตอนการดึงคำสั่งและถอดรหัสคำสั่งอีกครั้งเพื่ออ่านคำสั่งถัดไปในโปรแกรม

รายละเอียดเพิ่มเติมของเนื้อหาในส่วนนี้ แสดงในภาคผนวก ข



รูปที่ 3.4 แผนภูมิแสดงขั้นตอนการทำงานของหน่วยประมวลผล

ในรูปที่ 3.5 มีการใช้สัญลักษณ์ที่แทนชื่อสถานะต่างๆ สัญลักษณ์ที่ใช้นี้แสดงในภาคผนวก ข ตารางที่ ข.1



รูปที่ 3.5 แผนภูมิสถานะของหน่วยประมวลผลนี้

3.5.1 กลุ่มสถานะที่อยู่ในขั้นตอนการดึงคำสั่งและถอดรหัสคำสั่ง

ขั้นตอนการดึงคำสั่งและถอดรหัสคำสั่ง (Instruction fetch & decode phase) เป็นขั้นตอนแรกในการทำงานแต่ละคำสั่งของหน่วยประมวลผล ในขั้นตอนนี้หน่วยประมวลผลจะอยู่ในสถานะดึงคำสั่งและถอดรหัสคำสั่ง (Instruction fetch & decode state, ST_IFD) ในสถานะนี้หน่วย

ประมวลผลจะทำการอ่านคำสั่งจากหน่วยความจำตำแหน่งที่ค่าของรีจิสเตอร์ PC ชี้อยู่มาเก็บไว้ในรีจิสเตอร์ IR พร้อมทั้งเพิ่มค่าที่เก็บในรีจิสเตอร์ PC เพื่อให้ค่านี้ชี้ที่คำสั่งถัดไป

ในขณะเดียวกันคำสั่งที่อ่านมาได้นั้นจะถูกทำการถอดรหัส (Decode) เพื่อส่งต่อการทำงานไปยังกลุ่มสถานะที่อยู่ในขั้นตอนการกระทำที่ 1 (Execute1 phase) ต่อไป ซึ่งเมื่อหมดรอบนาฬิกาหน่วยประมวลผลจะเข้าสู่การทำงานในขั้นตอนการกระทำที่ 1 (Execute1 phase) ซึ่งคำสั่งที่อ่านได้ในขั้นตอนนี้จะเป็นตัวกำหนดสถานะของหน่วยประมวลผลในขั้นตอนถัดไป รายละเอียดแสดงภาคผนวก ข ตารางที่ ข.2

3.5.2 กลุ่มสถานะที่อยู่ในขั้นตอนการกระทำที่ 1

หลังจากที่ได้ทำการถอดรหัสคำสั่ง (Instruction decode) ไปแล้วในขั้นตอนการดึงคำสั่งและถอดรหัสคำสั่ง (Instruction fetch & decode phase) ที่ผ่านมา หน่วยประมวลผลจะเข้าสู่ขั้นตอนการกระทำที่ 1 (Execute1 phase) ในขั้นตอนนี้ประกอบด้วยสถานะทั้งหมด 15 สถานะ ดังแสดงในรูปที่ 3.5 พฤติกรรมการทำงานแสดงในภาคผนวก ข ตารางที่ ข.3

คำสั่งที่มีการทำงานที่ซับซ้อน ได้แก่ LDX, STV, INC, DEC และ IJX ไม่สามารถกระทำ (Execute) ได้เสร็จภายในหนึ่งรอบนาฬิกา (Clock cycle) หน่วยประมวลผลจะเข้าสู่ขั้นตอนการกระทำที่ 2 (Execute2 phase) เพื่อกระทำคำสั่งนั้นๆให้เสร็จ

สำหรับคำสั่งอื่นๆที่การทำงานไม่ซับซ้อน สามารถทำงานเสร็จได้ในหนึ่งรอบนาฬิกา หน่วยประมวลผลจะเข้าสู่ขั้นตอนการดึงคำสั่งและถอดรหัสคำสั่ง (Instruction fetch & decode phase) เพื่อเริ่มการทำงานในคำสั่งต่อไป

3.5.3 กลุ่มสถานะที่อยู่ในขั้นตอนการกระทำที่ 2

ขั้นตอนนี้จะเกิดขึ้นก็ต่อเมื่อคำสั่งที่หน่วยประมวลผลกำลังกระทำ (Execute) อยู่นั้นเป็นคำสั่ง LDX, STV, INC, DEC หรือ IJX ซึ่งจำเป็นต้องใช้ 2 รอบนาฬิกาในการทำงาน ในขั้นตอนนี้ประกอบด้วยสถานะทั้งหมด 4 สถานะ ดังแสดงในรูปที่ 3.5 พฤติกรรมการทำงานแสดงในภาคผนวก ข ตารางที่ ข.3 เมื่อทำงานในขั้นตอนนี้เสร็จหน่วยประมวลผลจะเข้าสู่ขั้นตอนการดึงคำสั่งและถอดรหัสคำสั่ง (Instruction fetch & decode phase) ดังแสดงในรูปที่ 3.4 และรูปที่ 3.5