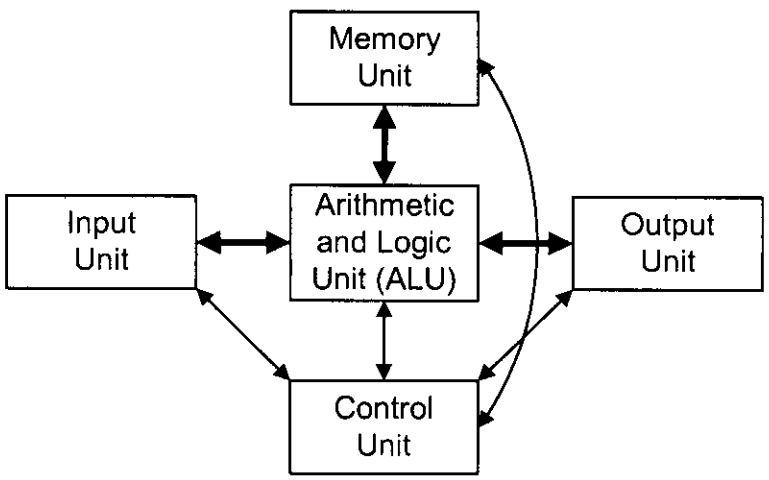
บทที่ 2

องค์ประกอบและการดำเนินงานของคอมพิวเตอร์

ปัจจุบันคอมพิวเตอร์ได้พัฒนาไปอย่างต่อเนื่องและรวดเร็ว แต่องค์ประกอบหลักในการทำงานยังคงมีพื้นฐานมาจากแนวคิดเดียวกันไม่ว่าจะเป็นคอมพิวเตอร์จากบริษัทผู้ผลิตรายใดก็ตาม ในบทนี้ จะกล่าวถึง นิยามของคำศัพท์ต่าง ๆ พร้อมทั้งหลักการพื้นฐานต่าง ๆ ที่เกี่ยวข้องกับคอมพิวเตอร์ เพื่อความเข้าใจที่ตรงกันในการอ่านเนื้อหาบทต่อๆไป

2.1 องค์ประกอบของคอมพิวเตอร์

องค์ประกอบและการติดต่อดำเนินงานของเครื่องคอมพิวเตอร์ตามรูปแบบของวอน นอยแมน (von Neumann model) ประกอบด้วยส่วนประกอบหลัก ๆ 5 ส่วน คือ หน่วยรับเข้า (input unit) , หน่วยความจำ (memory unit) , หน่วยคำนวณเลขคณิตและตรรกะ (arithmetic and logic unit : ALU) , หน่วยส่งออก (output unit) และ หน่วยควบคุม (control unit) ดังแสดงในภาพประกอบ 2.1



ภาพประกอบ 2.1 ส่วนประกอบหลักของคอมพิวเตอร์

[Murdocca, Miles J. and Heuring, Vincent P. (2000)]

จากส่วนประกอบหลักทั้งห้าส่วนจะมีสองส่วนที่อยู่ในหน่วยประมวลผลกลาง (central processor unit) หรือซีพียู (CPU) คือหน่วยคำนวณเลขคณิตและตรรกะ ทำหน้าที่คำนวณทางเลข

คณิตและดำเนินงานทางตรรกะ และหน่วยควบคุม ทำหน้าที่ควบคุมการทำงานของหน่วยอื่น ๆ ในระบบ นอกจากนี้ภายในซีพียูยังมีส่วนประกอบที่สำคัญที่ใช้ในการประมวลผลของซีพียู คือ รีจิสเตอร์ (registers) ซึ่งทำหน้าที่เก็บข้อมูลที่ซีพียูกำลังใช้ในการประมวลผลอยู่ รีจิสเตอร์แบ่งเป็น 2 ประเภท คือ

1. รีจิสเตอร์อเนกประสงค์ (general purpose register) เป็นรีจิสเตอร์ที่ผู้ใช้สามารถนำไปใช้ในการเขียนโปรแกรมได้ ปกติซีพียูจะมีรีจิสเตอร์ประเภทนี้จำนวนหนึ่ง หากในซีพียูมีรีจิสเตอร์ประเภทนี้มากจะทำให้การทำงานของซีพียูจะทำงานได้เร็วขึ้น ตัวอย่างของรีจิสเตอร์ประเภทนี้ เช่น แอคคิวมูเลเตอร์ (accumulator)

2. รีจิสเตอร์เฉพาะประสงค์ (special purpose register) เป็นรีจิสเตอร์ที่ระบบนำมาใช้ในการทำงานเฉพาะอย่าง เช่น

- PC (Program Counter) หรือ IP (Instruction Pointer) หรือ IC (Instruction Counter) เป็นรีจิสเตอร์ที่บรรจุเลขที่ตำแหน่งความจำหลักที่เป็นคำสั่งถัดไปที่ซีพียูจะดำเนินการประมวลผลหลังจากเสร็จสิ้นการประมวลผลคำสั่งปัจจุบัน
- IR (Instruction Register) เป็นรีจิสเตอร์ที่บรรจุคำสั่งปัจจุบันที่ซีพียูกำลังจะดำเนินงานประมวลผล
- MAR (Memory Address Register) เป็นรีจิสเตอร์ที่บรรจุเลขที่ตำแหน่งความจำหลักที่ซีพียูจะทำงานด้วย (การอ่านหรือบันทึก)
- MBR (Memory Buffer Register) หรือ MDR (Memory Data Register) เป็นรีจิสเตอร์ที่ใช้คู่กับ MAR เสมอ ในการอ่านหรือบันทึกข้อมูลของซีพียู โดยเป็นที่เก็บข้อมูลที่จะถูกบันทึก (ในการบันทึกข้อมูล) หรือเก็บข้อมูลที่ถูกอ่านมาจากความจำหลัก (ในการอ่าน)
- PSW (Program Status Word Register) เป็นรีจิสเตอร์ที่ประกอบด้วยบิตซึ่งใช้บอกสถานะต่าง ๆ ของการดำเนินงานของซีพียู เช่น มีความผิดพลาด (error) หรือไม่ ค่ารีจิสเตอร์ที่ใช้ในการดำเนินงานเป็นศูนย์หรือไม่ โดย PSW จะถูกใช้งานโดยแบ่งเป็นรายบิต

หน่วยความจำ ประกอบด้วย ความจำหลัก (main memory หรือ primary memory) และความจำรอง (secondary memory) ความจำหลัก หรือที่เรียกกันว่า แรม (RAM หรือ Random Access Memory) ทำหน้าที่เก็บคำสั่งและข้อมูลที่ซีพียูกำลังดำเนินงานด้วย ความจำรอง คือ หน่วยความจำที่ใช้เก็บข้อมูลและโปรแกรมได้ถาวร เช่น แถบบันทึก (tape) จานบันทึก (disk) และ แผ่นซีดี เป็นต้น

หน่วยรับเข้า ทำหน้าที่รับข้อมูลจากอุปกรณ์ภายนอกเพื่อนำมาประมวลผล ซึ่งตัวอย่างของอุปกรณ์ที่เป็นหน่วยรับเข้านี้ ได้แก่ แป้นพิมพ์ (keyboard) เมาส์ (mouse) ไมโครโฟน (microphone) เป็นต้น

หน่วยส่งออก ทำหน้าที่ส่งข้อมูลที่ได้จากการประมวลผลไปยังอุปกรณ์ภายนอก ได้แก่ จอภาพ (monitor) ลำโพง (speaker) และเครื่องพิมพ์ (printer) เป็นต้น อุปกรณ์บางอย่างทำหน้าที่ได้ทั้งหน่วยรับเข้าและหน่วยส่งออกในตัวเดียวกันได้ เช่น หน่วยขับแถบบันทึก (tape drive) หน่วยขับจานบันทึก (disk drive) เป็นต้น

หลักการทำงานพื้นฐานของคอมพิวเตอร์ตามแนวคิดของวอน นอยแมน คือหลักการที่เรียกว่า stored-program concepts นั่นคือคอมพิวเตอร์จะต้องนำส่วนที่เป็นโปรแกรมและส่วนที่เป็นข้อมูลมาเก็บไว้ในความจำหลักก่อนที่จะประมวลผล โดยซีพียูจะอ่านคำสั่งจากเลขที่ความจำหลักจากตำแหน่งที่ระบุมาทำการแปลว่าเป็นคำสั่งใดจากนั้นจะทำงานตามคำสั่งนั้นโดยมีหน่วยควบคุมทำการควบคุมและประสานการทำงานกับหน่วยอื่น ๆ เพื่อให้การทำงานเป็นไปอย่างถูกต้อง

ในการทำงานของคอมพิวเตอร์จำเป็นต้องมีการส่งผ่านสัญญาณต่าง ๆ ระหว่างหน่วยต่าง ๆ ภายในคอมพิวเตอร์ เรียกว่าบัส (bus) ดังแสดงในภาพประกอบที่ 2.2 โดยทั่วไปมีบัสอยู่ 3 ประเภท คือ

1. **Address bus** เป็นบัสที่ทำหน้าที่ส่งเลขที่ตำแหน่งความจำหลัก (memory address) จากซีพียูไปยังความจำหลัก
2. **Data bus** เป็นบัสที่ทำหน้าที่ส่ง/รับข้อมูล ซึ่งอาจเป็นคำสั่งหรือข้อมูลที่จะใช้ในการประมวลผล
3. **Control bus** เป็นบัสที่ทำหน้าที่ส่ง/รับสัญญาณควบคุมระหว่างซีพียูกับหน่วยอื่น ๆ เพื่อให้สามารถทำงานประสานกันได้อย่างราบรื่นและถูกต้อง

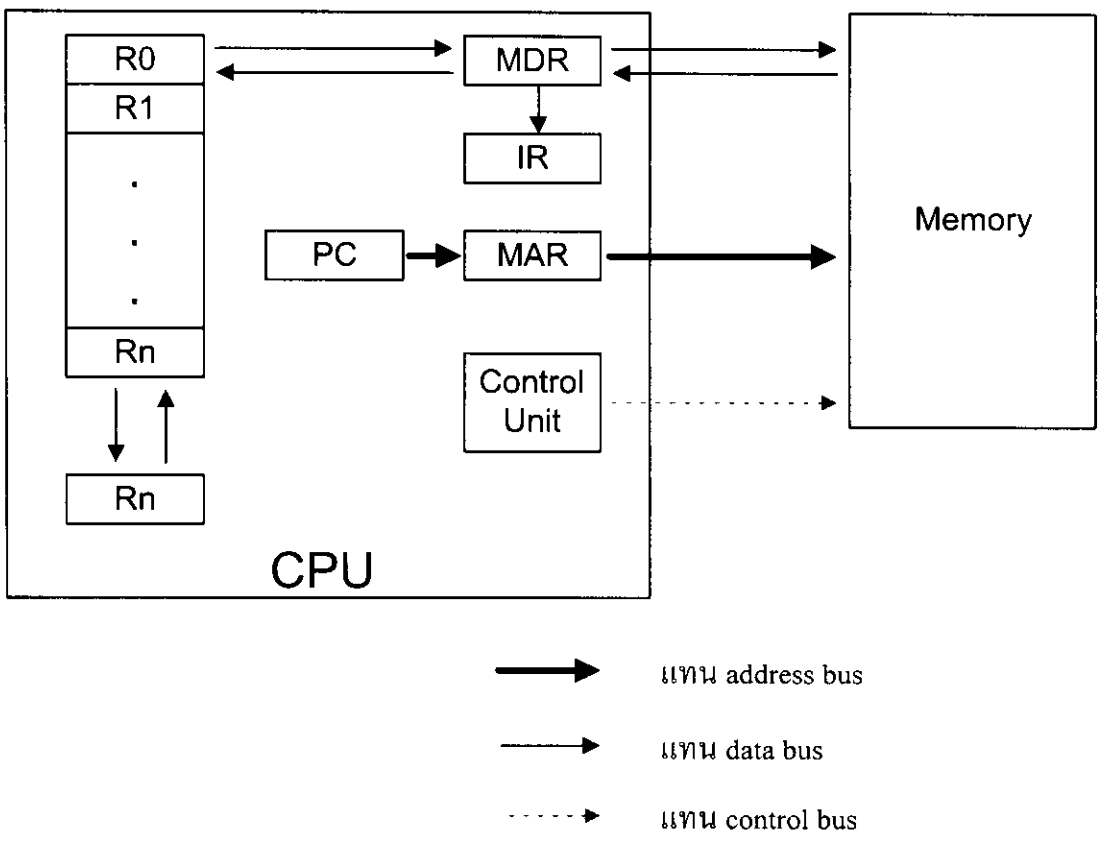
2.2 การดำเนินงานของคอมพิวเตอร์

การดำเนินงานของคอมพิวเตอร์ในการประมวลผลคำสั่งแต่ละคำสั่งของคอมพิวเตอร์จะมีสถานะการดำเนินงาน 2 ช่วงเรียกว่า fetch phase และ execution phase โดยที่คอมพิวเตอร์เมื่อถูกสั่งให้ประมวลผลโปรแกรมใด ๆ จะมีขั้นตอนการดำเนินงานที่หน่วยควบคุมจะต้องจัดลำดับการดำเนินงานตามขั้นตอนต่อไปนี้

```
Repeat
        Fetch phase
        Execute phase
Until halt or fatal error
```

ขั้นตอนในช่วง fetch phase เริ่มตั้งแต่ที่ซีพียูอ่านคำสั่งจากความจำหลักนำมาเก็บในรีจิสเตอร์ IR เพื่อส่งต่อให้กับวงจรถอดรหัสทำการแปลว่าคำสั่งนี้เป็นคำสั่งที่ต้องการให้ซีพียูดำเนินการอะไร หลังจากนั้นจะเป็นขั้นตอนช่วง execute phase ซึ่งเป็นขั้นตอนที่หน่วยควบคุมส่งสัญญาณควบคุมหน่วยอื่น ๆ ให้ดำเนินงานตามที่ต้องการ

การดำเนินงานประมวลผลคำสั่งของคอมพิวเตอร์ ซีพียูจำเป็นต้องทำงานประสานกันกับความจำหลักตลอดเวลา โดยโปรแกรมและข้อมูลที่จะประมวลผลต้องอยู่ในความจำหลัก และเริ่มต้นการประมวลผลโปรแกรมด้วยคำสั่งที่เก็บในตำแหน่งความจำหลักที่ถูกระบุในรีจิสเตอร์ PC หน่วยควบคุมจะส่งค่าจากรีจิสเตอร์ PC ไปยังรีจิสเตอร์ MAR และจากนั้นจะมีวงจรถอดรหัสเลขที่ความจำหลักที่เก็บคำสั่งที่จะนำมาประมวลผล คำสั่งที่อยู่ในความจำหลักตำแหน่งที่ถูกระบุจะถูกส่งผ่าน data bus ไปเก็บในรีจิสเตอร์ MDR และส่งต่อไปยังรีจิสเตอร์ IR เป็นการสิ้นสุดช่วง fetch phase หลังจากนั้นเป็นการดำเนินงานในช่วง execute phase ซึ่งจะดำเนินงานตามคำสั่งที่ถูกถอดรหัสได้ต่อไป



ภาพประกอบ 2.2 การทำงานประสานกันระหว่างซีพียูกับหน่วยความจำหลักในช่วง fetch phase

2.3 ประเภทและรูปแบบของคำสั่ง

ในการทำงานของระบบเครื่องคอมพิวเตอร์ คำสั่งและข้อมูลในความจำหลักที่เป็นภาษาเครื่อง (machine language) อยู่ในรูปแบบของบิต (bit pattern) จะถูกนำมาประมวลผลในซีพียู เป็นความยุ่งยากหากให้ผู้เขียนโปรแกรมเขียนโปรแกรมด้วยภาษาเครื่อง ภาษาแอสเซมบลีจึงเป็นภาษาระดับต่ำที่เหมาะสำหรับให้ผู้เขียนโปรแกรมนำมาเขียนโปรแกรมสั่งงานให้คอมพิวเตอร์ทำงานตามที่ผู้เขียนโปรแกรมต้องการ โดยเฉพาะผู้ที่กำลังศึกษาการทำงานของระบบเครื่องคอมพิวเตอร์ ภาษาแอสเซมบลีจึงเหมาะเป็นอย่างยิ่งในการศึกษาการทำงานของระบบเครื่องคอมพิวเตอร์ คำสั่งภาษาแอสเซมบลีหนึ่งคำสั่งจะถูกแปลเป็นคำสั่งภาษาเครื่องได้หนึ่งคำสั่งที่สมนัยกันเสมอ ประเภทของคำสั่งพื้นฐานที่จำเป็นต้องมีในระบบเครื่องคอมพิวเตอร์ต่าง ๆ ได้แก่

1. คำสั่งทางเลขคณิต เช่น คำสั่งบวก (addition) ลบ (subtraction) คูณ (multiplication) และหาร (division) และคำสั่งทางตรรกะ เช่น AND OR และ NOT

2. คำสั่งโอนย้ายข้อมูล เช่น

- store เป็นคำสั่งที่นำค่าในรีจิสเตอร์ของซีพียูไปเก็บในความจำหลัก
- load เป็นคำสั่งที่นำค่าในความจำหลักไปเก็บในรีจิสเตอร์ของซีพียู
- move เป็นคำสั่งที่นำค่าในเลขที่ความจำหลักตำแหน่งหนึ่งไปยังอีกตำแหน่งหนึ่ง

3. คำสั่งเปลี่ยนลำดับการทำงาน เช่น

- branch, jump, skip เป็นคำสั่งเปลี่ยนลำดับการทำงานไปยังคำสั่งในความจำหลักตำแหน่งใหม่แทนที่จะเป็นคำสั่งในความจำหลักตำแหน่งถัดไปต่อจากคำสั่งปัจจุบันที่กำลังประมวลผลอยู่ ซึ่งสามารถกำหนดให้มีการเปลี่ยนลำดับการประมวลผลแบบไม่มีเงื่อนไขหรือแบบมีเงื่อนไข

4. คำสั่งการดำเนินงานเคลื่อนย้ายบิตภายในเวิร์ด เช่น

- shift เป็นคำสั่งดำเนินการเลื่อนข้อมูลเป็นรายบิต เช่น เลื่อนข้อมูลไปทางซ้ายหรือขวาของเวิร์ดครั้งละหนึ่งบิต
- rotate เป็นคำสั่งดำเนินการหมุนข้อมูลเป็นรายบิต โดยถ้าเป็นการหมุนทางขวาข้อมูลบิตทางขวาสุดจะมาอยู่เป็นข้อมูลทางซ้ายสุดของเวิร์ด แต่ถ้าเป็นการหมุนทางซ้าย ข้อมูลบิตทางซ้ายสุดจะมาอยู่เป็นข้อมูลทางขวาสุดของเวิร์ด

5. คำสั่งรับเข้าข้อมูลและส่งออกผลลัพธ์ เช่น

- input เป็นคำสั่งที่นำเข้าข้อมูล
- output เป็นคำสั่งที่ส่งออกผลลัพธ์

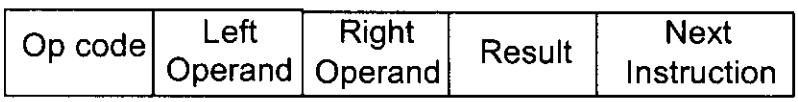
รูปแบบของคำสั่งโดยพื้นฐานแล้วจะประกอบด้วย 2 ส่วน ดังแสดงในภาพประกอบ 2.3 คือ ส่วนรหัสแทนการดำเนินการ (opcode) เป็นส่วนที่ระบุการดำเนินงาน ถ้าส่วนนี้มีจำนวน m บิต แสดงว่าสามารถมีคำสั่งที่เป็นไปได้สูงสุด 2^m คำสั่ง และส่วนรายละเอียดการดำเนินการ (operand) เป็นส่วนประกอบที่ทำให้คำสั่งนี้สมบูรณ์ เช่น การอ้างอิงตำแหน่งข้อมูลในความจำหลัก เป็นต้น ซึ่งขึ้นอยู่กับผู้ออกแบบระบบเครื่องคอมพิวเตอร์ว่าต้องการให้มีรายละเอียดอะไรบ้าง

ภาพประกอบ 2.3 รูปแบบทั่วไปของคำสั่งภาษาเครื่อง

รูปแบบคำสั่งการดำเนินหลักทางคำนวณและตรรกะของภาษาแอสเซมบลีสำหรับคอมพิวเตอร์แต่ละเครื่องจะมีรูปแบบแตกต่างกันแล้วแต่ผู้ออกแบบซีพียู ตามหลักการอาจมีรูปแบบคำสั่งได้ดังต่อไปนี้

1. Four-address format

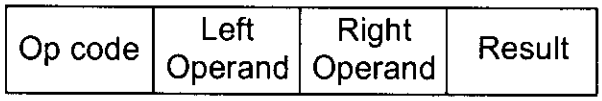
รูปแบบนี้เป็นรูปแบบคำสั่งการคำนวณและตรรกะที่ประกอบไปด้วยตัวถูกกระทำ (operand) ที่มีการใช้ความจำหลัก 4 ตำแหน่ง คือ ตัวถูกกระทำทางซ้ายและขวา (left operand และ right operand) ของการดำเนินงาน ตัวเก็บผลลัพธ์ (result) และตัวเก็บเลขที่ความจำหลักของคำสั่งถัดไปที่ซีพียูจะประมวลผล (next instruction) ดังแสดงในภาพประกอบ 2.4 รูปแบบนี้มีข้อเสีย คือ แต่ละคำสั่งจำเป็นต้องใช้จำนวนบิตมาก ทำให้เปลืองพื้นที่ในการใช้ความจำหลัก



ภาพประกอบ 2.4 รูปแบบคำสั่งชนิด Four-address format

2. Three-address format

รูปแบบคำสั่งนี้ประกอบไปด้วยตัวถูกกระทำที่ประกอบในคำสั่งมีจำนวน 3 ตัว ดังแสดงในภาพประกอบ 2.5 โดยตัดส่วนที่เป็นเลขที่ความจำหลักของคำสั่งถัดไปที่ซีพียูจะประมวลผลด้วย โดยต้องเพิ่มรีจิสเตอร์เพื่อระบุตำแหน่งของความจำหลักที่เก็บคำสั่งถัดไป ซึ่งมีข้อดีกว่ารูปแบบ four-address ในส่วนของการใช้เนื้อที่ความจำหลัก เนื่องจากจำนวนบิตที่ใช้แทนคำสั่งน้อยลง ทำให้ใช้เนื้อที่ในการแทนคำสั่งในความจำหลักน้อยลง รูปแบบคำสั่งแบบนี้มีใช้ในซีพียูยุคต้น ๆ



ภาพประกอบ 2.5 รูปแบบคำสั่งชนิด Three -address format

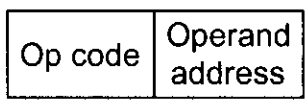
3. Two-address format

รูปแบบคำสั่งแบบนี้ประกอบไปด้วยตัวถูกกระทำในคำสั่งมีจำนวน 2 ตัว ดังแสดงในภาพประกอบ 2.6 โดยลดตัวถูกกระทำจากรูปแบบ three-address format ในส่วนของตัวเก็บผลลัพธ์ 1 ตัว ซึ่งตัวถูกกระทำในส่วนของตัวตั้งหรือตัวดำเนินการต้องทำหน้าที่เป็นตัวเก็บผลลัพธ์ด้วย จะทำให้จำนวนบิตที่นำไปใช้แทนในแต่ละคำสั่งมีจำนวนน้อยลง การใช้เนื้อที่ในความจำหลักมีประสิทธิภาพมากขึ้น คอมพิวเตอร์ส่วนใหญ่นิยมใช้รูปแบบนี้ เช่น ไมโครโพรเซสเซอร์ของบริษัทอินเทล ใช้รูปแบบคำสั่งแบบนี้โดยผลลัพธ์จากการคำเนินงานจะถูกเก็บที่ตัวถูกกระทำทางซ้าย

ภาพประกอบ 2.6 รูปแบบคำสั่งชนิด Two -address format

4. One-address format

รูปแบบคำสั่งแบบนี้ประกอบไปด้วยตัวถูกกระทำในคำสั่งมีจำนวน 1 ตัว ดังแสดงในภาพประกอบ 2.7 โดยจะต้องมีรีจิสเตอร์พิเศษช่วยในการทำงาน เช่น แอคคิวมูเลเตอร์ โดยแอคคิวมูเลเตอร์ทำหน้าที่เป็นตัวถูกกระทำในส่วนตัวตั้งหรือตัวดำเนินการ และตัวเก็บผลลัพธ์ เนื่องจากมีจำนวนตัวถูกกระทำเพียง 1 ตัว ทำให้ในแต่ละคำสั่งใช้จำนวนบิตน้อยกว่ารูปแบบ two-address format ตัวอย่างซีพียูที่ใช้รูปแบบนี้ เช่น PDP-8



ภาพประกอบ 2.7 รูปแบบคำสั่งชนิด One -address format

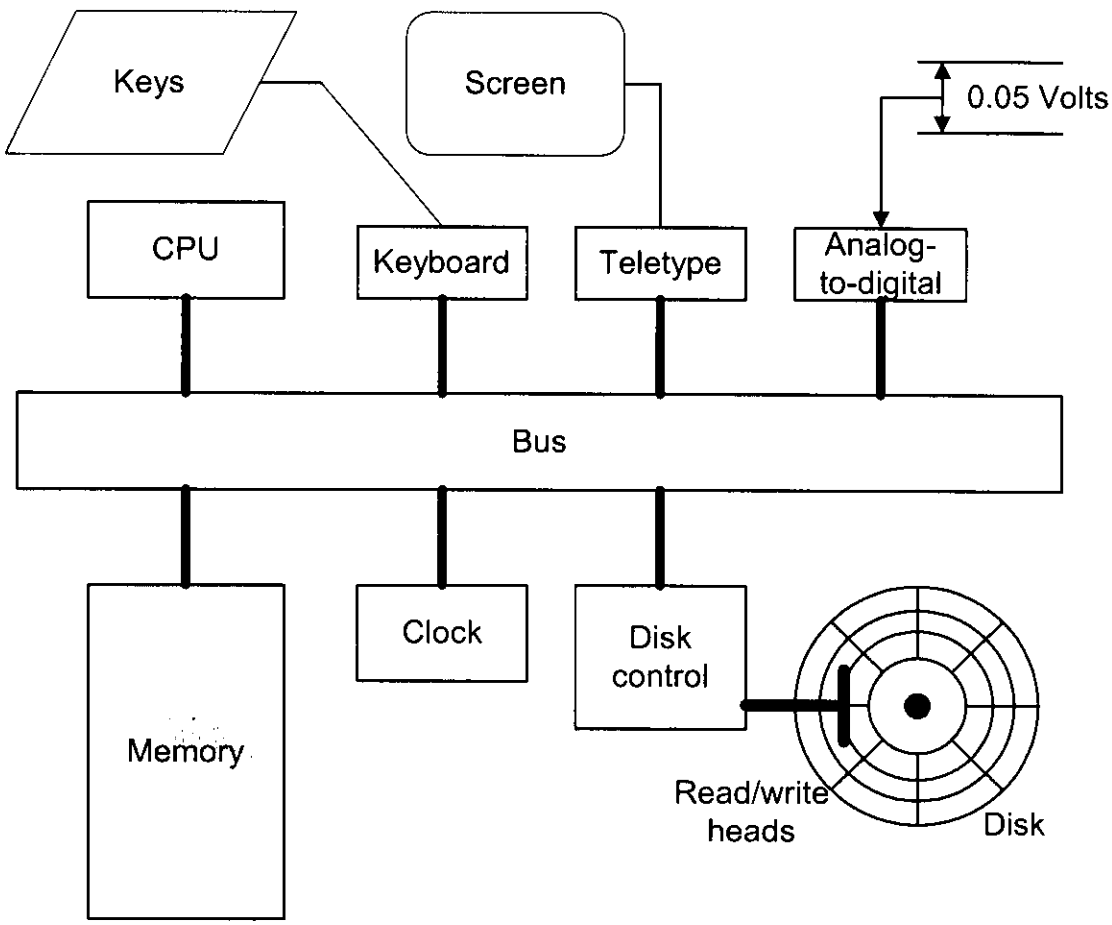
5. Zero-address format

รูปแบบคำสั่งแบบนี้ไม่มีตัวถูกกระทำในคำสั่ง ในการดำเนินงานจึงจำเป็นต้องใช้ system stack ช่วย ทำให้ใช้พื้นที่ในความจำหลักในการแทนแต่ละคำสั่งน้อยที่สุดเมื่อเทียบกับรูปแบบอื่น ๆ

2.4 การตรวจเอกสาร

การดำเนินงานพัฒนาโปรแกรมจำลองการทำงานของระบบคอมพิวเตอร์ส่วนใหญ่จะอยู่ในแวดวงการศึกษาเพื่อใช้เป็นเครื่องมือสนับสนุนการเรียนการสอนในวิชาที่เกี่ยวข้องกับองค์ประกอบและการดำเนินงานของระบบคอมพิวเตอร์ทั้งสิ้น เช่น

1. **The Simul8 Simulator.** (N. A. B. Gray , 1987-1990) เป็นโปรแกรมจำลองการทำงานของระบบคอมพิวเตอร์จากเครื่อง PDP-8 โครงสร้างในการจำลองประกอบด้วย bus, CPU, memory, keyboard, Teletype, analog-to-digital converter, disk, fixed-frequency clock ดังแสดงในภาพประกอบ 2.8



ภาพประกอบ 2.8 โครงสร้างของคอมพิวเตอร์ที่จำลองโดย Simul8

Simul8 ถูกพัฒนาด้วยภาษา pascal และ ภาษา C ซึ่งในส่วนของตัวแปลภาษาแอสเซมบลี (assembler) ถูกพัฒนา เป็นแบบ two pass assembler

Simul8 สามารถใช้ได้กับเครื่องคอมพิวเตอร์บนระบบปฏิบัติการ 3 รูปแบบ คือ เครื่องไมโครคอมพิวเตอร์ Macintosh เครื่องไมโครคอมพิวเตอร์ซึ่งใช้ระบบปฏิบัติการ DOS และเครื่องคอมพิวเตอร์ซึ่งใช้ระบบปฏิบัติการ UNIX

2. **PDP8/e emulator**. (Bill Haygood , 1993) เป็นโปรแกรมจำลองการทำงานของระบบคอมพิวเตอร์ PDP-8 เช่นกัน โดยโปรแกรมจำลองการทำงานนี้ถูกพัฒนาขึ้นด้วยภาษา C และทำงานภายใต้ระบบ AMIGA หรือ ระบบปฏิบัติการ MS-DOS คำสั่งในการสั่งให้โปรแกรมทำงานจะเป็นบรรทัดคำสั่งเช่นเดียวกับการเรียกคำสั่งต่าง ๆ ในระบบปฏิบัติการ DOS

3. **JFraCE.** (http://jfrace.sourceforge.net , 2001) เป็นกลุ่มงานที่ร่วมกันพัฒนาโปรแกรมจำลองการทำงานของระบบคอมพิวเตอร์ระบบเก่า เพื่อนำมาจำลองบนเครื่องคอมพิวเตอร์ในระบบปัจจุบัน โดยใช้ภาษา JAVA ในการพัฒนาโปรแกรม JFraCE จะเป็น Open Source โดยจะจำลองการทำงานในส่วนของ CPU , ความจำหลักและอุปกรณ์รอบข้าง โปรแกรมจำลองนี้ได้จำลองการทำงานของเครื่องคอมพิวเตอร์ต่าง ๆ เช่น Zuse's Z3, ENIAC, IBM 360, PDP 11 , ไมโครคอมพิวเตอร์ที่ใช้ CPU 8080 และ Apple II

4. **JavaScript PDP-8 Simulator.** (Neal Sample , 1997) เป็นโปรแกรมจำลองการทำงานของระบบคอมพิวเตอร์ PDP-8 เช่นกัน โดยพัฒนาด้วย JavaScript ทำให้สามารถใช้โปรแกรมจำลองการทำงานนี้ได้จาก web browser

5. **SIMH.** (Bob Supnik , 2001) เป็นกลุ่มของโปรแกรมจำลองการทำงานของระบบคอมพิวเตอร์ในหลาย ๆ ระบบ เช่น PDP-1, PDP-4, PDP-7, PDP-8, PDP-9, PDP-10, PDP-11, PDP-15, VAX, IBM 1401, IBM System 3, IBM 1130 เป็นต้น พัฒนาด้วยภาษา C โดยสามารถนำโปรแกรมจำลองการทำงานต่าง ๆ เหล่านี้ไปใช้กับระบบปฏิบัติการต่าง ๆ เช่น Windows 9x/NT/2000, DEC UNIX, Open VMS, Linux, NetBSD, Solaris, OS/2, Macintosh OS 9 และ Macintosh OS X

6. **PDP-8 Simulator–Web Editor** (http://www.cit.gu.edu.au/teaching/1507CIT/assignment2/pdp8.htm , 2003) เป็นโปรแกรมจำลองการทำงานระบบคอมพิวเตอร์ PDP-8 ที่ทำงานได้จาก web browser โปรแกรมจำลองการทำงานนี้เขียนด้วยภาษาจาวาที่ทำงานในลักษณะของ applet สามารถใช้จำลองการทำงานตามภาษาแอสเซมบลีได้ แต่โปรแกรมจำลองการทำงานนี้มีข้อจำกัดคือ ไม่สนับสนุนคำสั่งประเภทติดต่อกับอุปกรณ์ภายนอก ตัวแปลภาษาแอสเซมบลีไม่

สนับสนุนการผสมคำสั่งประเภท operate และในการใช้งานจะต้องใช้งานที่ Internet Explorer 3.0 หรือ Netscape 4.0 หรือสูงกว่า

2.5 PDP-8

เครื่องคอมพิวเตอร์ PDP-8 เป็นคอมพิวเตอร์ขนาดกลางของบริษัท DEC (Digital Equipment Corporation) มีขนาดของคำสั่งและข้อมูลที่ใช้ในการประมวลผลแต่ละครั้ง จำนวน 12 บิต ทำให้สามารถแทนรูปแบบบิตของข้อมูลได้ 4096 รูปแบบ คือตั้งแต่รูปแบบบิตที่มีค่า 0 ถึง 4095 เมื่อใช้ระบบเลขฐานแปดแทนจะมีค่าตั้งแต่ 0 ถึง 7777 นอกจากนี้ เครื่องคอมพิวเตอร์ PDP-8 มีเลขที่ความจำหลักทั้งหมด 4096 ตำแหน่ง ดังนั้นรีจิสเตอร์ PC และ MAR ของ PDP-8 จึงถูกออกแบบให้มีขนาด 12 บิต

เครื่องคอมพิวเตอร์ PDP-8 มีจำนวนรีจิสเตอร์ที่ผู้เขียนโปรแกรมสามารถใช้ได้ 2 ตัว โดยเป็นรีจิสเตอร์ที่มีขนาด 12 บิต 1 ตัว คือ ACC (accumulator) และรีจิสเตอร์ขนาด 1 บิต 1 ตัว คือ link

ความจำหลักของ PDP-8 จะแบ่งพื้นที่ความจำหลักออกเป็นหน้า (page) จำนวน 32 หน้า ๆ ละ 128 เวิร์ด (word) หรือ 200_8 เวิร์ด โดยเริ่มหมายเลขหน้าตั้งแต่ 0_8 ถึง 37_8 แต่ละหน้าถูกกำหนดให้มีการใช้งานต่าง ๆ ดังแสดงในตาราง 2.1

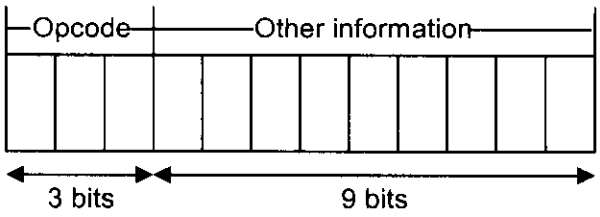
ตาราง 2.1 รายละเอียดของการกำหนดการใช้งานความจำหลัก

หมายเลขหน้า	เลขที่ความจำหลัก (เลขฐานแปด)	การใช้งาน
0	0000-0177	เก็บข้อมูลส่วนกลาง (Global page)
1	0200-0377	โปรแกรมหลัก
2	0400-0577	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
3	0600-0777	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
4	1000-1177	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
5	1200-1377	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
6	1400-1577	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
7	1600-1777	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
10	2000-2177	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
11	2200-2377	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
12	2400-2577	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ

ตาราง 2.1 รายละเอียดของการกำหนดการใช้งานความจำหลัก (ต่อ)

หมายเลขหน้า	เลขที่ความจำหลัก (เลขฐานแปด)	การใช้งาน
13	2600-2777	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
14	3000-3177	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
15	3200-3377	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
16	3400-3577	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
17	3600-3777	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
20	4000-4177	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
21	4200-4377	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
22	4400-4577	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
23	4600-4777	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
24	5000-5177	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
25	5200-5377	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
26	5400-5577	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
27	5600-5777	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
30	6000-6177	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
31	6200-6377	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
32	6400-6577	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
33	6600-6777	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
34	7000-7177	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
35	7200-7377	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
36	7400-7577	ซับรูทีน หรือ ข้อมูลที่ต้องการใช้งานอื่น ๆ
37	7600-7777	โปรแกรมโหลดเดอร์

คำสั่งภาษาแอสเซมบลีของ PDP-8 มีคำสั่งหลักเพียง 8 คำสั่ง มีรูปแบบในการจัดการคำสั่งภาษาเครื่องดังแสดงในภาพประกอบ 2.9 โดย 3 บิตแรกเป็นรหัสแทนตัวดำเนินการ ซึ่งมีอยู่ 8 ตัว ดำเนินการ ดังแสดงในตาราง 2.2



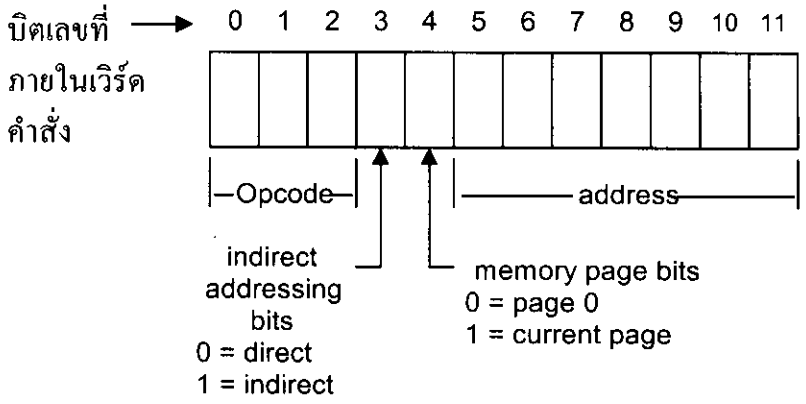
ภาพประกอบ 2.9 รูปแบบของคำสั่งภาษาเครื่อง PDP-8

ตาราง 2.2 ตัวดำเนินการหลักของ PDP-8

ลำดับ	รูปแบบบิต	สัญลักษณ์	แทนการดำเนินการ
1	000	and	logical and operation
2	001	tad	two's complement add operation
3	010	isz	increment and skip if zero
4	011	dca	deposite and clear accumulator
5	100	jms	a subroutine call instruction
6	101	jmp	a jump (go to) instruction
7	110	iot	input/output instructions
8	111	opr	data manipulation and all kinds of test instructions

คำสั่งต่าง ๆ ที่ใช้ใน PDP-8 สามารถจัดแบ่งเป็นกลุ่มคำสั่งออกเป็น 3 กลุ่ม คือ

1. **Memory reference instructions** เป็นกลุ่มคำสั่งที่มีการอ้างอิงเลขที่ความจำหลักในการดำเนินงาน ตามตาราง 2.2 คือ ลำดับที่ 1 – 6 ประกอบด้วยคำสั่ง and, tad, isz, dca, jms และ jmp รายละเอียดรูปแบบบิตของคำสั่งแสดงในภาพประกอบ 2.10 โดย 3 บิตแรก (บิตเลขที่ 0 – 2) ภายในเวิร์ด จะแทนรหัสตัวดำเนินการของคำสั่ง บิตเลขที่ 3 แทนการอ้างอิงเลขที่ความจำหลักแบบ direct/indirect บิตเลขที่ 4 แทนการอ้างอิงเลขที่ความจำหลักว่าเป็นการอ้างอิงเลขที่ความจำหลักในหน้าใด บิตเลขที่ 5 ถึง 11 แทนเลขที่ความจำหลักที่ใช้ประกอบในคำสั่ง



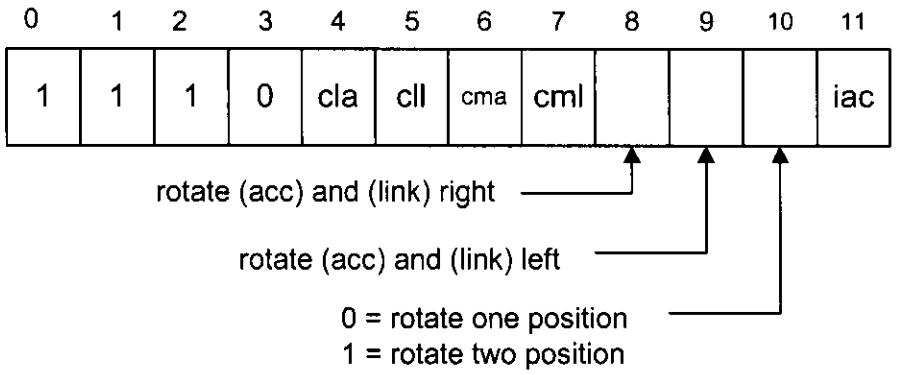
ภาพประกอบ 2.10 รายละเอียดโครงสร้างรูปแบบของคำสั่งกลุ่ม Memory reference instructions

ในการประมวลผลคำสั่งในกลุ่ม memory reference instructions จะต้องมีการหาเลขที่ความจำหลักจริงที่ใช้ในการประมวลผล (Effective address) โดยพิจารณาบิตเลขที่ 3, 4 ประกอบกับบิตเลขที่ 5 ถึง 11 ซึ่งสามารถพิจารณาได้ดังนี้

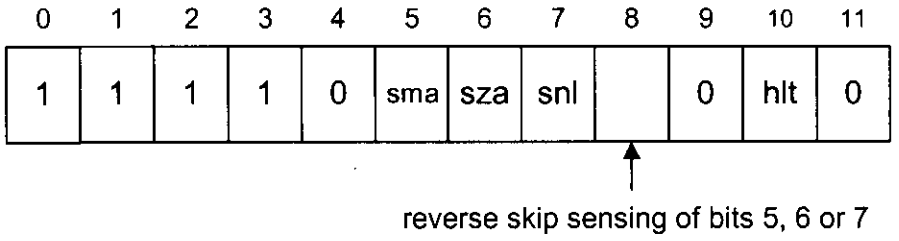
- กรณีบิตเลขที่ 3 และ 4 มีค่าเป็น 0 เป็นการอ้างอิงเลขที่ความจำหลัก ซึ่งใช้เก็บข้อมูลที่ใช้ประมวลผลคำสั่งเป็นแบบ direct และอยู่ในหน้าความจำหลักที่ 0 เลขที่ความจำหลักที่ใช้จริง คือ ค่าซึ่งแทนด้วยรูปแบบบิตจากบิตเลขที่ 5 ถึง 11 ในคำสั่ง
- กรณีบิตเลขที่ 3 มีค่าเป็น 0 และบิตเลขที่ 4 มีค่าเป็น 1 เป็นการอ้างอิงเลขที่ความจำหลัก ซึ่งใช้เก็บข้อมูลที่ใช้ประมวลผลคำสั่งเป็นแบบ direct แต่เลขที่ความจำหลักที่อ้างถึงอยู่ในหน้าความจำหลักปัจจุบันที่คำสั่งอยู่ เลขที่ความจำหลักที่ใช้จริง คือ ค่าที่แทนด้วยรูปแบบบิต ซึ่งเกิดจากการเอาบิตเลขที่ 0 ถึง 4 ของรีจิสเตอร์ PC ตามด้วยรูปแบบบิตจากบิตเลขที่ 5 ถึง 11 ของรีจิสเตอร์ IR
- กรณีบิตเลขที่ 3 มีค่าเป็น 1 และบิตเลขที่ 4 มีค่าเป็น 0 เป็นการอ้างอิงเลขที่ความจำหลัก ซึ่งใช้เก็บข้อมูลที่ใช้ประมวลผลคำสั่งเป็นแบบ indirect โดยเลขที่ความจำหลักที่ใช้เก็บเลขที่ความจำหลักที่ใช้จริงได้จากการแทนค่ารูปแบบบิตจากบิตเลขที่ 5 ถึง 11 ในคำสั่ง
- กรณีบิตเลขที่ 3 มีค่าเป็น 1 และบิตเลขที่ 4 มีค่าเป็น 1 เป็นการอ้างอิงเลขที่ความจำหลัก ซึ่งใช้เก็บข้อมูลที่ใช้ประมวลผลคำสั่งเป็นแบบ indirect โดยเลขที่ความจำหลักที่ใช้เก็บเลขที่ความจำหลักที่ใช้จริงอยู่ในหน้าความจำหลักปัจจุบันที่คำสั่งอยู่ ซึ่งต้องหาได้จากการแทนด้วยรูปแบบบิต ซึ่งเกิดจากการเอาบิตเลขที่ 0 ถึง 4 ของรีจิสเตอร์ PC ตามด้วยรูปแบบบิตจากบิตเลขที่ 5 ถึง 11 ของรีจิสเตอร์ IR

2. **Operate instruction** เป็นคำสั่งลำดับที่ 8 ตามตาราง 2.2 เกี่ยวข้องกับการดำเนินงานต่าง ๆ กับข้อมูลในรีจิสเตอร์ ACC และ link แบ่งได้เป็น 2 กลุ่ม

- Group 1 : data manipulation เป็นกลุ่มคำสั่งที่ดำเนินงานข้อมูลในรีจิสเตอร์ ACC และ link ประกอบด้วยคำสั่ง nop, iac, ral, rar, rtl, rtr, cml, cma, cll และ cla รายละเอียดโครงสร้างของคำสั่งดังแสดงในภาพประกอบ 2.11 รายละเอียดการดำเนินงานของคำสั่งดังแสดงในตาราง 2.3
- Group 2 : all kinds of tests เป็นกลุ่มคำสั่งที่ดำเนินงานทดสอบผลลัพธ์จากการดำเนินงานในรีจิสเตอร์ ACC และ link ประกอบด้วยคำสั่ง hlt, skp, snl, szl, sza, sna, sma และspa รายละเอียดโครงสร้างของคำสั่งดังแสดงในภาพประกอบ 2.12 รายละเอียดการดำเนินงานของคำสั่งดังแสดงในตาราง 2.4



ภาพประกอบ 2.11 โครงสร้างของคำสั่ง Operate กลุ่ม 1



ภาพประกอบ 2.12 โครงสร้างของคำสั่ง Operate กลุ่ม 2

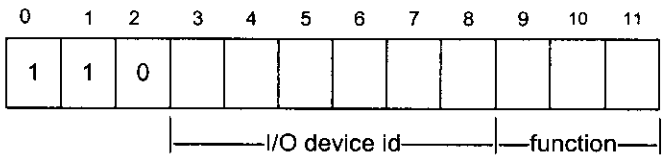
ตาราง 2.3 คำสั่งในกลุ่ม Operate instruction : data manipulation ของ PDP-8

รูปแบบบิต (ในรูปฐานแปด)	**สัญลักษณ์**	**แทนการดำเนินการ**
7000	nop	no operation
7001	iac	increment (acc)
7004	ral	rotate (acc) and (link) left one position
7010	rar	rotate (acc) and (link) right one position
7006	rtl	rotate (acc) and (link) left two position
7012	rtr	rotate (acc) and (link) right two position
7020	cml	complement (link)
7040	cma	complement (ac)
7100	cll	clear (link)
7200	cla	clear (acc)

ตาราง 2.4 คำสั่งในกลุ่ม Operate instruction : all kinds of tests ของ PDP-8

รูปแบบบิต (ในรูปฐานแปด)	**สัญลักษณ์**	**แทนการดำเนินการ**
7402	hlt	halt the program
7410	skp	skip the next instruction
7420	snl	skip on non-zero (link)
7430	szl	skip if zero (link)
7440	sza	skip if (acc) = 0
7450	sna	skip if (acc) $\neq$ 0
7500	sma	skip if (acc) $<$ 0
7510	spa	skip if (acc) $\geq$ 0

3. Input/output instruction เป็นคำสั่งลำดับที่ 7 ตามตาราง 2.2 เกี่ยวข้องกับการดำเนินงานรับเข้าข้อมูลและส่งออกผลลัพธ์ รายละเอียดโครงสร้างของคำสั่งดังแสดงในภาพประกอบ 2.13 รายละเอียดการดำเนินงานของคำสั่งแสดงในตาราง 2.5



ภาพประกอบ 2.13 โครงสร้างของคำสั่ง Input/output

ตาราง 2.5 คำสั่งในกลุ่ม Input/output instruction ของ PDP-8

รูปแบบบิต (ในรูปฐานแปด)	**สัญลักษณ์**	**แทนการดำเนินการ**
6036	krb	read data from keyboard buffer to acc, clear keyboard flag
6031	ksf	skip if keyboard flag set
6042	tcf	teletype clear flag
6044	tpc	teletype print character
6046	tls	teletype load and send (clear flag and print) tls = tcf + tpc
6041	tsf	skip if teletype flag set
6501	clkcf	clear clock flag
6502	clksf	skip if clock flag set
6504	clkt	clock toggle
6601	adcrb	a/d read data buffer & clear flag
6602	adsf	skip if a/d flag set
6604	adstrt	start a/d sampling
6002	iof	interrupts off (disabled)
6001	ion	interrupts on (enabled)
6400	dlsk	load disk block register and start seek
6401	dssf	skip if disk flag set
6402	dscf	clear disk seek flag
6410	dlma	load disk memory address register (from acc)
6411	drd	start reading from disk into memory
6412	dtsf	skip if disk data-transfer-complete- flag is set
6414	dtcf	clear disk data-transfer-complete-flag
6415	dwrt	start writing to disk from memory