บทที่ 4

การออกแบบ

เนื่องจากซีพียู 8 บิตมีโครงสร้างหลัก และหลักการทำงานที่เหมือนกัน มีชุดคำสั่งและแอดเดรสซิงโหมดที่คล้ายกัน ถ้าเรามีเซ็ตของส่วนประกอบของซีพียู และเซ็ตของคำสั่งเซ็ตหนึ่งที่ครอบคลุมลักษณะการทำงานของซีพียู 8 บิต หรือสามารถสังเคราะห์ลักษณะการทำงานแบบต่างๆ ขึ้นมาได้จากเซ็ตๆ นี้ และมีกลไกที่สามารถจำลองซีพียูขึ้นมาจากส่วนประกอบและคำสั่งที่เราเลือกขึ้นมาจากเซ็ตดังกล่าวได้ เราจะสามารถจำลองซีพียู 8 บิตที่มีลักษณะใดๆได้ตามแต่ความต้องการ จะไม่จำกัดอยู่เพียงซีพียูเบอร์ใดเบอร์หนึ่ง จากโปรแกรมจำลองซีพียู 6502 (สมภพ คำนูณเศรษฐ์ และสุริยัน ติษยาธิคม, 2532) ซึ่งเป็น Specific CPU Simulator เราอาจพิจารณาว่า Resources ของโปรแกรมจำลองซีพียูนี้เป็น Specific Resources ถ้าเราเปลี่ยนแปลงให้เป็น Generic Resources และเพิ่มกลไก ซึ่งอนุญาตให้ผู้ใช้โปรแกรมจำลองการทำงานซีพียู เป็นผู้กำหนด Specific Resources ขึ้นจาก Generic Resources ที่มีอยู่ด้วยตนเอง โปรแกรมจำลองการทำงานซีพียู ก็จะมีลักษณะเป็นแบบ Generic ได้ คือสามารถจำลองซีพียูอย่างไม่ตายตัว

Generic Resources ได้แก่

1) Generic Registers
2) Generic Addressing Modes
3) Generic Instruction Set
4) Generic Assembler

การออกแบบ Generic Registers

โครงสร้างแบบ Generic ประกอบด้วย รีจิสเตอร์ขนาด 8 และ 16 บิต จำนวนหนึ่งซึ่งมากพอที่จะนำมากำหนดใช้งานเป็น Data Register และ Address Register ภายในซีพียูได้ เช่น Accumulator, Index Register ตามความต้องการของการจำลองซีพียู 8 บิต ซึ่งอยู่ภายในขอบเขตอันหนึ่ง

การออกแบบ Generic Addressing Modes

คำสั่งต่างๆ ของซีพียู มีความสัมพันธ์กับลักษณะของการแอดเดรสหน่วยความจำ ในการทำงานเกี่ยวกับการจัดการข้อมูลของซีพียู ซีพียูจะต้องรู้จักวิธีการไปนำข้อมูลมาจากหน่วยความจำเพื่อจัดการตามคำสั่ง แต่ละซีพียู มีแอดเดรสชิงโหมดแตกต่างกันบ้างตามชื่อที่เรียกหรือจำนวน แต่โดยวิธีการแล้วมีความคล้ายคลึงกัน และแอดเดรสชิงโหมดที่ครอบคลุมลักษณะการแอดเดรสของซีพียู 8 บิต มีเพียงไม่กี่วิธี เราสามารถกำหนดแอดเดรสชิงโหมดชุดนี้ขึ้นมา และกำหนดให้กลไกของการทำงานตามคำสั่งของซีพียูที่จำลองขึ้นสามารถรู้จัก และแอ็กเซสข้อมูลได้ตามแอดเดรสชิงโหมดชุดนี้ ในการจำลอง Specific CPU ให้เลือกแอดเดรสชิงโหมด มาจากแอดเดรสชิงโหมดชุดนี้

การออกแบบ Generic Instruction Set

ลักษณะของชุดคำสั่งที่แตกต่างกันของซีพียูแต่ละเบอร์ อยู่ที่ Mnemonic และ Opcode ที่กำหนดขึ้นคู่กัน และจำนวนคำสั่ง ส่วนในการทำงานอย่างเดียวกันก็ไม่มีความแตกต่างกัน ดังนั้นเราอาจออกแบบคำสั่งขึ้นมาชุดหนึ่งซึ่งครอบคลุมการทำงานต่างๆของซีพียู 8 บิตเรียกว่า Generic Instruction Set โดยในการจำลอง Specific CPU ให้เลือกคำสั่งขึ้นมาจากคำสั่งชุดนี้ หรือสังเคราะห์ขึ้นโดยการโปรแกรมคำสั่งต่างๆ ที่มีอยู่ในชุดคำสั่งนี้ให้เป็นคำสั่งของ Specific CPU ใดๆ ตามที่ต้องการ ในเรื่องเกี่ยวกับคำสั่ง เมื่อวิเคราะห์ลึกลงไปถึงการทำงานในระดับ Machine Level เราสามารถนำแนวความคิดของ Risc (Reduced Instructions Set Computer) มาประยุกต์ใช้ ด้วยหลักการ 2 ประการคือ

-การจัดการข้อมูลกระทำในรีจิสเตอร์เท่านั้น

-คำสั่ง Load / Store เท่านั้นที่สามารถแอ็กเซสหน่วยความจำ

การนำมาประยุกต์กับการออกแบบคำสั่งของ Generic CPU ทำให้เรากำหนดชุดคำสั่งที่ครอบคลุมการทำงานของซีพียู 8 บิต ได้ง่ายขึ้น โดยเราสามารถกำหนดคำสั่งขึ้นมากลุ่มหนึ่งซึ่งจัดการข้อมูลในรีจิสเตอร์เท่านั้นโดยไม่คำนึงถึงแอดเดรสชิงโหมด และกำหนดคำสั่งขึ้นอีกกลุ่มหนึ่ง ที่ทำงานเกี่ยวกับการแอ็กเซสหน่วยความจำด้วยแอดเดรสชิงโหมด ต่างๆ เมื่อนำคำสั่งในกลุ่มแรกมาจับคู่กับคำสั่งในกลุ่มที่สองก็จะทำให้สามารถสังเคราะห์คำสั่งสำหรับ Specific CPU ขึ้นได้หลากหลายรูปแบบ

คำสั่งจัดการข้อมูลในรีจิสเตอร์ ส่วนใหญ่ เป็นคำสั่งทำงานทางคณิตศาสตร์-ลอจิก ซึ่งแต่ละซีพียู แตกต่างกันน้อยมาก แม้แต่ซีพียู Z80 ที่มีคำสั่งทำงานที่ค่อนข้างซับซ้อน ก็ประกอบด้วยการทำงานพื้นฐานทั้งสิ้น เช่นคำสั่ง "LDI" ประกอบด้วยการทำงาน

(DE)--(HL) : ย้ายข้อมูลในหน่วยความจำจากตำแหน่งหนึ่งไปยังอีกตำแหน่งหนึ่ง

INC HL : เพิ่มหนึ่ง Pointer HL

DEC BC : ลดหนึ่ง Counter BC

ดังนั้นหลักที่ใช้ในการออกแบบคำสั่งกลุ่มแรกนี้ก็คือ รวบรวมจากคำสั่งของซีพียู 8 บิต โดยเลือกเอาคำสั่งประเภทที่ทำงานพื้นฐาน เนื่องจากคำสั่งที่ซับซ้อนกว่าสามารถสังเคราะห์ขึ้นได้จากคำสั่งพื้นฐาน

การดึงลักษณะร่วมพื้นฐานออกมาสร้างเป็น Basic Set นี้ เป็นกลไกสำคัญของ Generic Machine เพราะทำให้สามารถสังเคราะห์ลักษณะเฉพาะใดๆ ขึ้นมาได้จาก Combinations ต่างๆของ Basic Set แต่ Basic Set นั้นจะต้องครอบคลุมการทำงานพื้นฐานไว้ได้ทั้งหมด

<u>การออกแบบ Generic Assembler</u>

การทำงานของแอสเซมเบลอร์ทั่วไป กลไกที่แปลจาก Mnemonic เป็น Instruction Code นอกจากจะขึ้นกับตารางการแปลคำสั่งตามชุดคำสั่งของซีพียูแล้วยังขึ้นกับ Assembly Syntax ด้วย ที่ทำให้แอสเซมเบลอร์สามารถแยกแยะแอดเดรส-

ชิงโหมดได้ เนื่องจากเรากำหนดแอดเดรสชิงโหมดไว้ชุดเดียวที่สามารถใช้กำหนดแอดเดรสชิงโหมดของซีพียู 8 บิตใดๆ ได้ แต่ Syntax ของภาษาจริงมีได้ไม่จำกัด ดังนั้น เราจำเป็นต้องออกแบบให้มีการป้อนข้อมูลเกี่ยวกับ Assembly Syntax ของผู้ใช้ และให้แอสเซมเบลอร์ มีกลไกการทำงานที่สัมพันธ์กับข้อมูลนี้