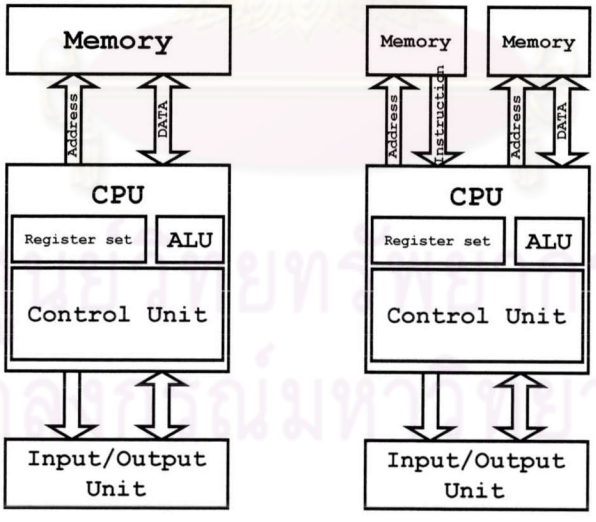
บทที่ 2
การออกแบบไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ถูกนำไปใช้งานด้านต่าง ๆ มากขึ้น เพราะระบบที่ใช้ไมโครคอนโทรลเลอร์สามารถปรับปรุงฟังก์ชันการทำงานได้ง่ายและรวดเร็ว เนื่องจากแก้ไขในส่วนของโปรแกรมควบคุมเท่านั้น ประกอบกับเทคโนโลยีวงจรรวมในปัจจุบันสามารถสร้างไมโครคอนโทรลเลอร์ที่มีความซับซ้อนและความเร็วในการประมวลผลได้มากขึ้น อย่างไรก็ตามงานในแต่ละด้านต้องการคุณสมบัติของไมโครคอนโทรลเลอร์ที่แตกต่างกัน เช่น ความสามารถในการประมวลผลที่ซับซ้อน ความเร็วในการทำงาน พลังงานที่ใช้ ขนาด และต้นทุนของไมโครคอนโทรลเลอร์ ดังนั้นสถาปัตยกรรมของหน่วยประมวลผลกลาง และอุปกรณ์บริวารในไมโครคอนโทรลเลอร์จึงมีความหลากหลาย เพื่อตอบสนองความต้องการใช้งานในด้านต่าง ๆ

2.1 สถาปัตยกรรมของฮาร์ดแวร์ [17]

สถาปัตยกรรมของฮาร์ดแวร์เป็นปัจจัยสำคัญตัวหนึ่งที่กำหนดคุณสมบัติของไมโครคอนโทรลเลอร์ สถาปัตยกรรมที่เป็นที่รู้จักกันดีคือ สถาปัตยกรรมแบบวอนนอยแมน (von Neumann Architecture)



รูปที่ 2.1 (ซ้าย) สถาปัตยกรรมแบบวอนนอยแมน (ขวา) สถาปัตยกรรมแบบฮาร์วาร์ด

สถาปัตยกรรมแบบวอนนอยแมนเป็นสถาปัตยกรรมอย่างง่าย องค์ประกอบหลักของไมโครคอนโทรลเลอร์แบบวอนนอยแมน คือ หน่วยความจำหลัก หน่วยประมวลผลกลาง และหน่วยอินพุต-เอาต์พุต ส่วนโปรแกรมควบคุมการทำงานถูกแปลงให้เป็นรหัสดำเนินการ (Operational Code: Op-code) และจัดเก็บไว้ในหน่วยความจำหลัก ซึ่งใช้สำหรับจัดเก็บข้อมูลสำหรับการประมวลผลในการทำงานด้วย

การทำงานแบ่งออกเป็นวงรอบการทำงาน (Machine cycle) ประกอบด้วยวงรอบเฟทช์คำสั่ง (Instruction Fetch) หน่วยประมวลผลกลางอ่านรหัสดำเนินการจากหน่วยความจำหลักทีละคำสั่ง และตีความรหัสดำเนินการ และวงรอบดำเนินการ (Execution) หน่วยประมวลผลกลางดำเนินการทำงานตามรหัสคำสั่งที่ได้รับ เมื่อดำเนินการเสร็จเรียบร้อยจึงเริ่มวงรอบเฟทช์คำสั่งของคำสั่งถัดไป บางไมโครคอนโทรลแบบวอนนอยแมนอาจมีวงรอบถอดรหัสคำสั่ง (Instruction Decode) แยกออกมาโดยเฉพาะ

หน่วยประมวลผลกลางประกอบด้วยส่วนสำคัญดังต่อไปนี้

หน่วยควบคุม (Control Unit) ทำหน้าที่ควบคุมส่วนอื่น ๆ ในหน่วยประมวลผล กลางให้ทำงานสอดคล้องกัน และเป็นไปตามรหัสดำเนินการที่ได้รับ

หน่วยคำนวณและตรรกะ (Arithmetic and Logic Unit: ALU) ทำหน้าที่ประมวลผลข้อมูล เช่น การคำนวณ ดำเนินการทางตรรก และการเลื่อนบิตข้อมูลเป็นต้น

ชุดรีจิสเตอร์ (Register set) กลุ่มของรีจิสเตอร์เพื่อเก็บข้อมูลที่ใช้ในการประมวลผล และเพื่อใช้งานเฉพาะด้าน

ตัวนับโปรแกรม (Program counter) เป็นรีจิสเตอร์ที่เก็บค่าตำแหน่งของคำสั่งในหน่วยความจำ ที่ถูกนำมาดำเนินการในคำสั่งถัดไป

สถาปัตยกรรมแบบฮาร์วาร์ด (Harvard Architecture) มีโครงสร้างคล้ายกับสถาปัตยกรรมแบบวอนนอยแมน (ดูรูปที่ 2.1) แต่หน่วยความจำแบ่งออกเป็นสองส่วน โปรแกรมและข้อมูลถูกจัดเก็บไว้คนละส่วน หน่วยประมวลผลกลางต้องมีบัส 2 ชุดเพื่อติดต่อกับหน่วยความจำทั้งสองโครงสร้างดังกล่าวช่วยให้หน่วยประมวลผลสามารถอ่านหรือเขียนหน่วยความจำทั้งสองชุดได้พร้อมกัน ซึ่งเป็นลักษณะเด่นของสถาปัตยกรรมแบบฮาร์วาร์ด

เมื่อพิจารณาโครงสร้างของหน่วยประมวลผลกลาง สามารถแยกออกเป็นสองส่วนสำคัญคือ ส่วนควบคุม (Control path) และ ส่วนดำเนินการ (Operation path) หรือเรียกอีกอย่างว่าส่วนข้อมูล (Data path) การใช้องค์ประกอบดังกล่าวมาเป็นเกณฑ์ในการแบ่งแยกประเภทของสถาปัตยกรรมของไมโครคอนโทรลเลอร์ แบ่งได้ดังนี้ (ดูรูปที่ 2.2)

- Single Instruction stream, Single Data stream (SISD) ไมโครคอนโทรล-เลอร์แบบวอนนอยแมนและฮาร์วาร์ด ก็จัดเป็นไมโครคอนโทรลเลอร์แบบนี้ เพราะสามารถดำเนินการตามคำสั่ง ณ เวลาใดเวลาหนึ่งได้เพียงคำสั่งเดียว และเป็นการดำเนินการกับข้อมูลเพียงชุดเดียว
- Single Instruction stream, Multiple Data stream (SIMD) มีส่วนควบคุมชุดเดียว และส่วนดำเนินการมีมากกว่า 1 ชุด ส่วนควบคุมส่งสัญญาณควบคุมส่วนดำเนินการ ดังนั้นโครงสร้างนี้สามารถดำเนินการคำสั่งแบบเดียวกันกับข้อมูลหลายชุดในเวลาเดียวกัน
- Multiple Instruction stream, Multiple Data stream (MIMD) ประกอบด้วยส่วนควบคุมและส่วนดำเนินการหลายชุด ส่วนควบคุมหนึ่งตัวควบคุมส่วนดำเนินการเพียงตัวเดียว ดังนั้นโครงสร้างนี้สามารถดำเนินการคำสั่งที่แตกต่างกันกับข้อมูลหลายชุดในเวลาเดียวกัน เรียกว่า Multiprocessor

2.2 สถาปัตยกรรมของชุดคำสั่ง [15,17]

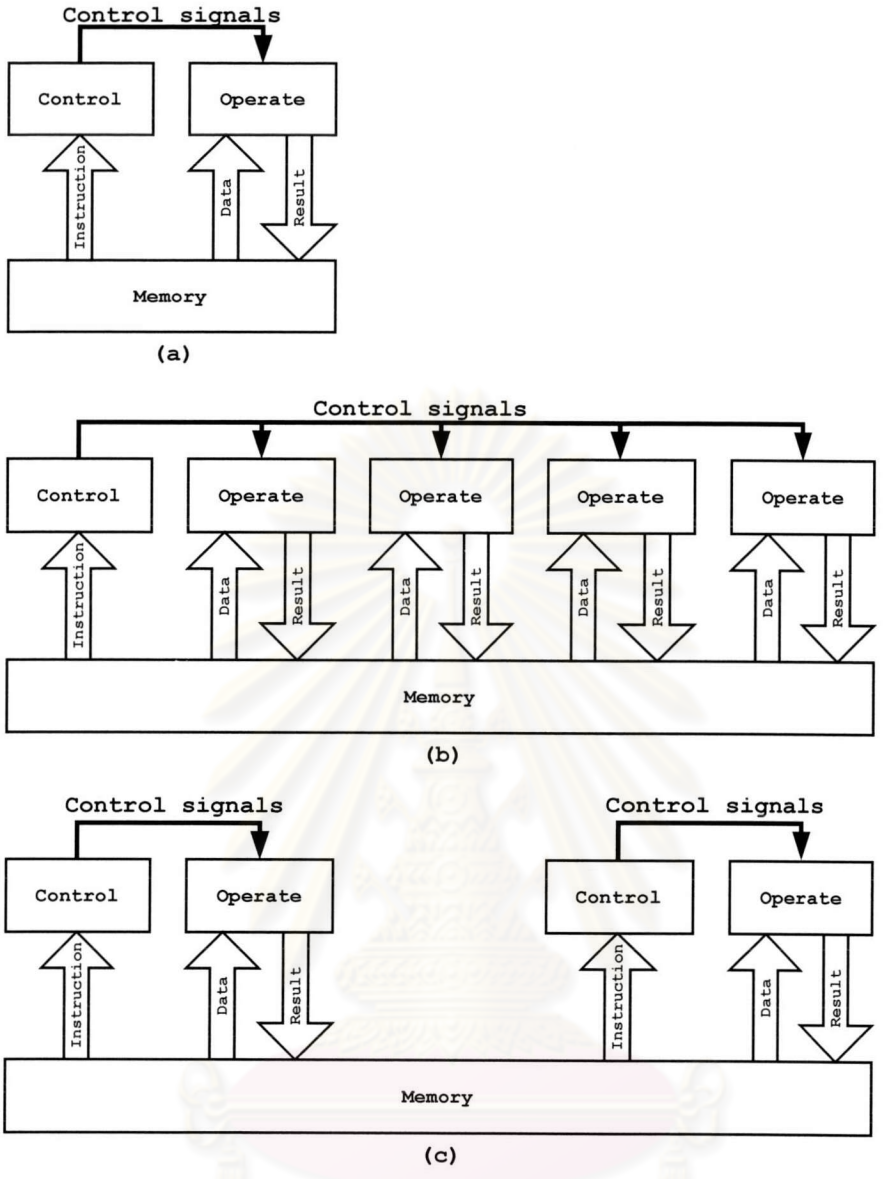
สถาปัตยกรรมของชุดคำสั่ง (Instruction-Set Architecture: ISA) เป็นส่วนที่สำคัญของไมโครคอนโทรลเลอร์ เพราะเป็นตัวกำหนดให้ผู้ใช้สามารถใช้งานทรัพยากรที่มีอยู่ในโครงสร้างทางฮาร์ดแวร์ได้ สถาปัตยกรรมของชุดคำสั่งแสดงให้ผู้ใช้ทราบถึง ชนิดข้อมูล (Data type) ชุดรีจิสเตอร์คำสั่ง และแอสเดรสซิ่งโหมด (Addressing mode)

2.2.1 ชนิดข้อมูล

สถาปัตยกรรมของชุดคำสั่งแสดงให้เห็น ข้อมูลที่หน่วยประมวลผลกลาง หรือ ตัวช่วยประมวลผลสามารถนำไปประมวลผลได้ เช่น จำนวนเต็ม (Integer) ทศนิยม (Floating point) เป็นต้น

2.2.2 ชุดรีจิสเตอร์

โครงสร้างทางฮาร์ดแวร์จะมีรีจิสเตอร์จำนวนมาก ซึ่งบางส่วนผู้ใช้ไม่สามารถเข้าใช้งานโดยตรงผ่านชุดคำสั่งที่มีอยู่ได้ เช่น รีจิสเตอร์คำสั่ง (Instruction Register) คือรีจิสเตอร์ที่ใช้เก็บรหัสดำเนินการ และมีรีจิสเตอร์พิเศษที่ชุดคำสั่งให้ผู้ใช้ใช้งานเป็นพิเศษ เช่น รีจิสเตอร์สถานะที่ประกอบด้วยบิตที่แสดงสถานะการประมวลผล (Processor-status bits)



รูปที่ 2.2 โครงสร้างของไมโครคอนโทรลเลอร์แบบต่าง ๆ [17] (a) SISD (b) SIMD (c) MIMD

เนื่องจากการเข้าถึงข้อมูลที่อยู่ในรีจิสเตอร์สามารถเข้าถึงได้เร็วกว่าข้อมูลที่อยู่ในหน่วยความจำ จึงควรออกแบบให้คำสั่งส่วนใหญ่สามารถใช้งานรีจิสเตอร์ได้ และรหัสที่ใช้เลือกรีจิสเตอร์ควรเลือกได้จำนวนสูงสุด เช่น ถ้ามี N บิตสำหรับเลือกรีจิสเตอร์ แล้วควรมีรีจิสเตอร์ที่ใช้งานได้ 2^N ตัวเป็นต้น แต่ในบางกรณีรูปแบบคำสั่งเป็นตัวกำหนดให้ผู้ใช้เรียกใช้รีจิสเตอร์แต่ละตัวต่างกัน ทั้งที่รีจิสเตอร์เหล่านั้นไม่ได้มีโครงสร้างทางฮาร์ดแวร์ที่แตกต่างกันเลย

2.2.3 รูปแบบคำสั่ง

ลักษณะของคำสั่งที่ดำเนินการกับข้อมูล มีการอ้างถึงตัวถูกดำเนินการ (Operand) และผลลัพธ์ แบ่งได้ 3 ประเภทคือ

- Memory-to-register instructions คำสั่งที่ตัวถูกดำเนินการอยู่ในหน่วยความจำ และ ใช้รีจิสเตอร์เก็บผลลัพธ์
- Memory-to-memory instructions คำสั่งที่ตัวถูกดำเนินการ และ ผลลัพธ์อยู่ในหน่วยความจำ
- Register-to-register instructions คำสั่งที่ใช้ตัวถูกดำเนินการ และ เก็บผลลัพธ์ เฉพาะรีจิสเตอร์เท่านั้น

ไมโครคอนโทรลเลอร์บางตัวมีคำสั่งดำเนินการข้อมูลแบบ register-to-register instructs ยกเว้นคำสั่ง LOAD และ STORE เท่านั้นที่ใช้อ้างถึงหน่วยความจำได้ เรียกชุดคำสั่งแบบนี้ว่า Load-and-store instruction sets [17]

รูปแบบคำสั่งยังกำหนดความยาวของรหัสดำเนินการ (Instruction Length) หมายถึงจำนวนบิตที่ใช้เข้ารหัสคำสั่ง ซึ่งการกำหนดความยาวของรหัสดำเนินการแบ่งได้เป็น

ความยาวรหัสดำเนินการแบบไม่คงที่ (Variable Code-Length) หมายถึงแต่ละคำสั่งใช้จำนวนบิตในการแทนคำสั่งไม่เท่ากัน ใช้จำนวนตำแหน่งในหน่วยความจำสำหรับจัดเก็บรหัสดำเนินการแตกต่างกันไป การเข้ารหัสคำสั่งแบบนี้ทำให้ความหนาแน่นการรหัส (Code density) สูง การใช้หน่วยความจำโปรแกรมเป็นไปอย่างมีประสิทธิภาพ แต่ทำให้กระบวนการเฟทช์คำสั่งมีความซับซ้อน

ความยาวรหัสดำเนินการแบบคงที่ (Fixed Code-Length) ทุกคำสั่งใช้รหัสดำเนินการมีความยาวเท่ากันทุกคำสั่ง และในกรณีที่รหัสดำเนินการของหนึ่งคำสั่งถูกเก็บไว้ในหน่วยความจำเพียงตำแหน่งเดียวเรียกว่า One-Word Instruction ซึ่งเหมาะสำหรับระบบที่ต้องการความเร็วในการประมวลผล [15]

2.2.4 ชุดคำสั่ง

คำสั่งเป็นการดำเนินการที่หน่วยประมวลผลกลางสามารถดำเนินการได้ โดยไมโครคอนโทรลเลอร์แต่ละตัวมีคำสั่งที่แตกต่างกัน แต่สามารถแบ่งเป็นกลุ่มคำสั่งได้ดังนี้

กลุ่มดำเนินการข้อมูล เช่น การคำนวณ การดำเนินการทางตรรกะ การเลื่อนบิตข้อมูล และการเปลี่ยนชนิดข้อมูล เป็นต้น

กลุ่มโอนย้ายข้อมูล เป็นคำสั่งเพื่อย้ายข้อมูลจากที่เก็บที่หนึ่งไปยังที่จัดเก็บที่ใหม่ ซึ่งเป็นรีจิสเตอร์ หรือหน่วยความจำ

กลุ่มควบคุม เป็นคำสั่งเกี่ยวกับการควบคุมการทำงานของไมโครโทรลเลอร์ เช่น คำสั่งกระโดดของโปรแกรม คำสั่งเปรียบเทียบและทดสอบเงื่อนไข เป็นต้น

กลุ่มอื่น ๆ เช่น คำสั่งเกี่ยวกับการจัดการอินพุต-เอาต์พุต คำสั่งเกี่ยวกับการขัดจังหวะ (Interrupt) เป็นต้น

2.2.5 แอสเดรสซิ่งโหมด

แอสเดรสซิ่งโหมด เป็นการกำหนดวิธีการอ้างถึงข้อมูลที่จัดเก็บอยู่ใน รีจิสเตอร์ และหน่วยความจำ หากแอสเดรสซิ่งโหมดมีหลายรูปแบบ ทำให้ผู้ใช้สามารถเขียนโปรแกรมได้กระชับ แต่การออกแบบฮาร์ดแวร์ก็มีความซับซ้อนมากขึ้นเช่นกัน ตัวอย่างแอสเดรสซิ่งโหมด เช่น แอสเดรสซิ่งโหมดแบบรีจิสเตอร์ แบบใช้ทันที แบบโดยตรง แบบโดยอ้อม ฯลฯ

2.2.6 การออกแบบชุดคำสั่ง

การออกแบบชุดคำสั่งให้มีประสิทธิภาพ ผู้ออกแบบชุดคำสั่งต้องคำนึงสิ่งต่อไปนี้

- ความสมบูรณ์ (Completeness) คือ ชุดคำสั่งต้องครอบคลุมการประมวลผลได้ครบตามความสามารถของฮาร์ดแวร์
- ไม่ซ้ำซ้อน (Orthogonality) คือ ในบรรดาชุดคำสั่งที่มี ต้องมีจำนวนคำสั่งที่ให้ผลการดำเนินการเหมือนกันต่ำ หรือ ไม่มีเลย
- ความสอดคล้อง (Compatibility) กับชุดคำสั่งของไมโครคอนโทรลเลอร์อื่นที่ผู้ใช้นิยมใช้มาก่อน ทำให้สามารถแปลงเป็นรหัสสำหรับไมโครคอนโทรลเลอร์ที่ออกแบบใหม่ได้ จึงเป็นแรงจูงใจให้ผู้ใช้เปลี่ยนมาใช้ไมโครคอนโทรลเลอร์ใหม่ได้ง่าย โดยที่ไม่ต้องพัฒนาโปรแกรมใหม่

2.3 ตัวประมวลผลแบบริสก์

ตัวประมวลผลแบบริสก์ (RISC: Reduced Instruction-Set Computer) เป็นสถาปัตยกรรมของคอมพิวเตอร์ที่เป็นพัฒนาขึ้นโดยอาศัยหลักการดังนี้

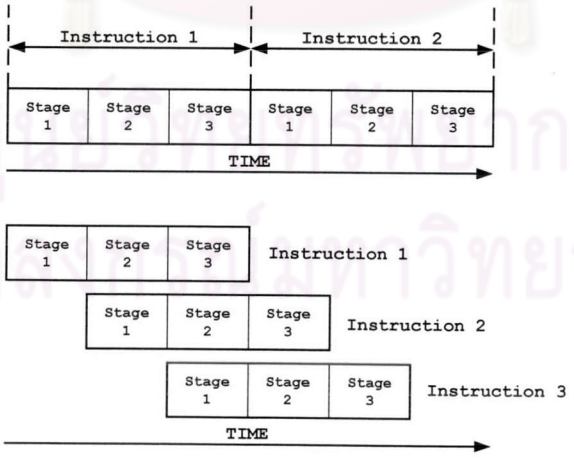
- มีจำนวนคำสั่งค่อนข้างน้อย เลือกเฉพาะการดำเนินการที่มีการใช้งานบ่อย ส่วนการดำเนินการที่ซับซ้อนที่ตัวประมวลผลแบบซิสก์(CISC: Complex Instruction-Set Computer) ดำเนินการได้นั้น อาศัยดำเนินการคำสั่งที่มีอยู่

หลายคำสั่งเพื่อให้ได้ผลลัพธ์ที่เหมือนกัน จึงกล่าวได้ว่าริสก์มีคำสั่งน้อย โครงสร้างทางฮาร์ดแวร์จึงมีความซับซ้อนต่ำ แต่ใช้คำสั่งจำนวนมากในการดำเนินให้เทียบเท่าซิสก์ ความซับซ้อนจึงอยู่กับการพัฒนาโปรแกรม จึงแตกต่างจากซิสก์มีชุดคำสั่งมากทำให้โครงสร้างทางฮาร์ดแวร์ซับซ้อน แต่เขียนโปรแกรมได้กระทัดรัดและมีประสิทธิภาพ

- ริสก์มีโครงสร้างทางฮาร์ดแวร์ไม่ซับซ้อนจึงทำงานได้เร็ว ชุดคำสั่งจึงหลีกเหลี่ยงการติดต่อหน่วยความจำเนื่องจากเป็นการลดความเร็วในการทำงาน ดังนั้นจึงจำเป็นต้องมีรีจิสเตอร์จำนวนมากไว้เพื่อเก็บข้อมูลระหว่างประมวลผล
- นิยมใช้งานไปป์ไลน์ (Pipeline) เพื่อเพิ่มความเร็วในการทำงาน

2.3.1 การใช้งานไปป์ไลน์

การใช้งานไปป์ไลน์เป็นเทคนิคที่ช่วยเพิ่มความเร็วในการทำงานที่มีลักษณะเป็นลำดับต่อเนื่อง (Sequential) ได้ดี โดยใช้หลักการแบ่งงานหนึ่งงานออกเป็นงานย่อย ๆ โดยมีวงจรสำหรับทำงานย่อยโดยเฉพาะ ตัวอย่างเช่น หน่วยประมวลกลางแบบที่ไม่ใช้โครงสร้างแบบไปป์ไลน์ดำเนินการคำสั่ง 1 คำสั่งใช้ 3 วงรอบการทำงาน แล้วจึงดำเนินการคำสั่งถัดไปได้ (ดูรูปที่ 2.3 บน) การใช้โครงสร้างไปป์ไลน์เข้าช่วยแบ่งการทำงานออกเป็น 3 ขั้นตอน (ดูรูปที่ 2.3 ล่าง) เมื่อวงจรที่ดำเนินการขั้นตอนแรกของคำสั่งที่ 1 เสร็จแล้วส่งให้กับวงจรในส่วนที่สองทำงานต่อไป และสามารถดำเนินการขั้นตอนแรกของคำสั่งที่ 2 ต่อไปได้โดยไม่ต้องรอให้ดำเนินการคำสั่งที่ 1 เสร็จเรียบร้อยก่อน เมื่อดำเนินการอย่างต่อเนื่อง 1 คำสั่งจะใช้เวลาเพียง 1 วงรอบการทำงานเท่านั้น



รูปที่ 2.3 (บน)ไม่ใช้โครงสร้างแบบไปป์ไลน์ (ล่าง)ใช้โครงสร้างแบบไปป์ไลน์

2.3.2 เทคนิคการออกตัวประมวลผลกลางแบบริสก์

เทคนิคการออกแบบตัวประมวลผลกลางแบบริสก์ เป็นข้อแนะนำสำหรับผู้ออกแบบตัวประมวลผลกลางเท่านั้น ผู้ออกแบบสามารถการออกแบบตัวประมวลกลางให้มีลักษณะแตกต่างออกไปได้ ขึ้นอยู่กับความต้องการของผู้ออกแบบและความเหมาะสมในการใช้งานเป็นหลัก

- ใช้งานโครงสร้างแบบไปป์ไลน์ เพื่อเพิ่มความเร็วในการถอดรหัสคำสั่งและดำเนินการประมวลผล
- ปกติไมโครคอนโทรลเลอร์แบบริสก์ ไม่สามารถปรับปรุงรหัสดำเนินการในหน่วยความจำได้ เพราะการยอมให้มีการปรับปรุงรหัสดำเนินการต้องใช้ฮาร์ดแวร์ช่วยตรวจสอบความถูกต้อง ทำให้โครงสร้างฮาร์ดแวร์ซับซ้อน
- สำหรับหน่วยประมวลผลกลางแบบริสก์ อาจใช้เทคนิคและฮาร์ดแวร์ช่วยให้การประมวลผลเร็วขึ้นเช่น การเฟทช์คำสั่งล่วงหน้า(Prefetch) คำสั่งที่ตำแหน่งที่เป็นเป้าหมายในคำสั่งกระโดด
- ใช้โครงสร้างแบบฮาร์วาร์ด เพราะสามารถช่วยป้องกันการปรับปรุงรหัสดำเนินการในหน่วยความจำได้
- มีจำนวนรีจิสเตอร์มาก เพื่อเก็บข้อมูลชั่วคราวทำให้สามารถลดการติดต่อระหว่างหน่วยประมวลผลกลางกับหน่วยความจำได้
- ออกแบบให้มีหน่วยถอดรหัสคำสั่งและหน่วยดำเนินการแยกออกจากกัน
- ในการกระโดดของโปรแกรม คำสั่งที่ตามหลังคำสั่งกระโดดเรียกว่า Branch-delay slot ริสก์โดยส่วนใหญ่จะดำเนินการคำสั่งที่ตามหลังคำสั่งกระโดดซึ่งเฟทช์มาแล้ว แม้ว่าจะมีการกระโดดของโปรแกรมก็ตาม หากผู้ใช้ไม่ต้องการให้คำสั่งหลังคำสั่งกระโดดต้องเติมคำสั่ง NOP (No Operation)
- เนื่องจากการติดต่อหน่วยความจำช้ากว่าการดำเนินการภายใน เพื่อลดเวลาที่เสียไปกับคำสั่ง LOAD คำสั่งที่ตามคำสั่ง LOAD เรียกว่า Load-delay slot หน่วยประมวลกลางจะดำเนินการคำสั่งเหล่านั้นขณะรอผลการดำเนินการคำสั่ง LOAD

2.4 คุณลักษณะของสถาปัตยกรรมไมโครคอนโทรลเลอร์

เนื่องจากสถาปัตยกรรมไมโครคอนโทรลเลอร์แต่ละแบบมีข้อดีข้อเสียที่แตกต่างกันขึ้นอยู่กับลักษณะงานที่นำไปใช้ด้วย การหาเกณฑ์ประเมินคุณภาพสถาปัตยกรรมไมโครคอนโทรลเลอร์จึงทำได้ไม่ชัดเจนนัก แต่พิจารณาคุณลักษณะของไมโครคอนโทรลเลอร์ได้จากหัวข้อเหล่านี้

- ขอบเขตของงานที่สามารถนำไมโครคอนโทรลเลอร์ประยุกต์ใช้งานได้ โดยทั่วไปไมโครคอนโทรลเลอร์ที่ใช้งานได้กว้างขวางมักจะมีชุดคำสั่งและแอสเดรสซิ่งโหมดมาก(โครงสร้างแบบซิสก์) ทำให้ความซับซ้อนในการออกแบบมากขึ้น
- ความสามารถของไมโครคอนโทรลเลอร์ในการประยุกต์ใช้งานเฉพาะด้านใดด้านหนึ่งได้ดี
- ประสิทธิภาพ โดยพิจารณาจาก จำนวนฮาร์ดแวร์ที่ถูกใช้งานในขณะดำเนินการปกติ หากมีจำนวนฮาร์ดแวร์ที่อยู่ในสภาวะว่างงานจำนวนน้อยถือว่ามีประสิทธิ-ภาพดี
- ความง่ายในการใช้งาน หมายถึงความง่ายในการพัฒนาโปรแกรมด้วยชุดคำสั่งของไมโครคอนโทรลเลอร์นั้น
- สามารถขยายสถาปัตยกรรมได้ หมายถึงสามารถพัฒนาสถาปัตยกรรมของทั้งฮาร์ดแวร์และชุดคำสั่งให้มีความสามารถประมวลผลมากขึ้น โดยที่โปรแกรมที่พัฒนาด้วยชุดคำสั่งเดิมยังสามารถใช้งานได้อยู่