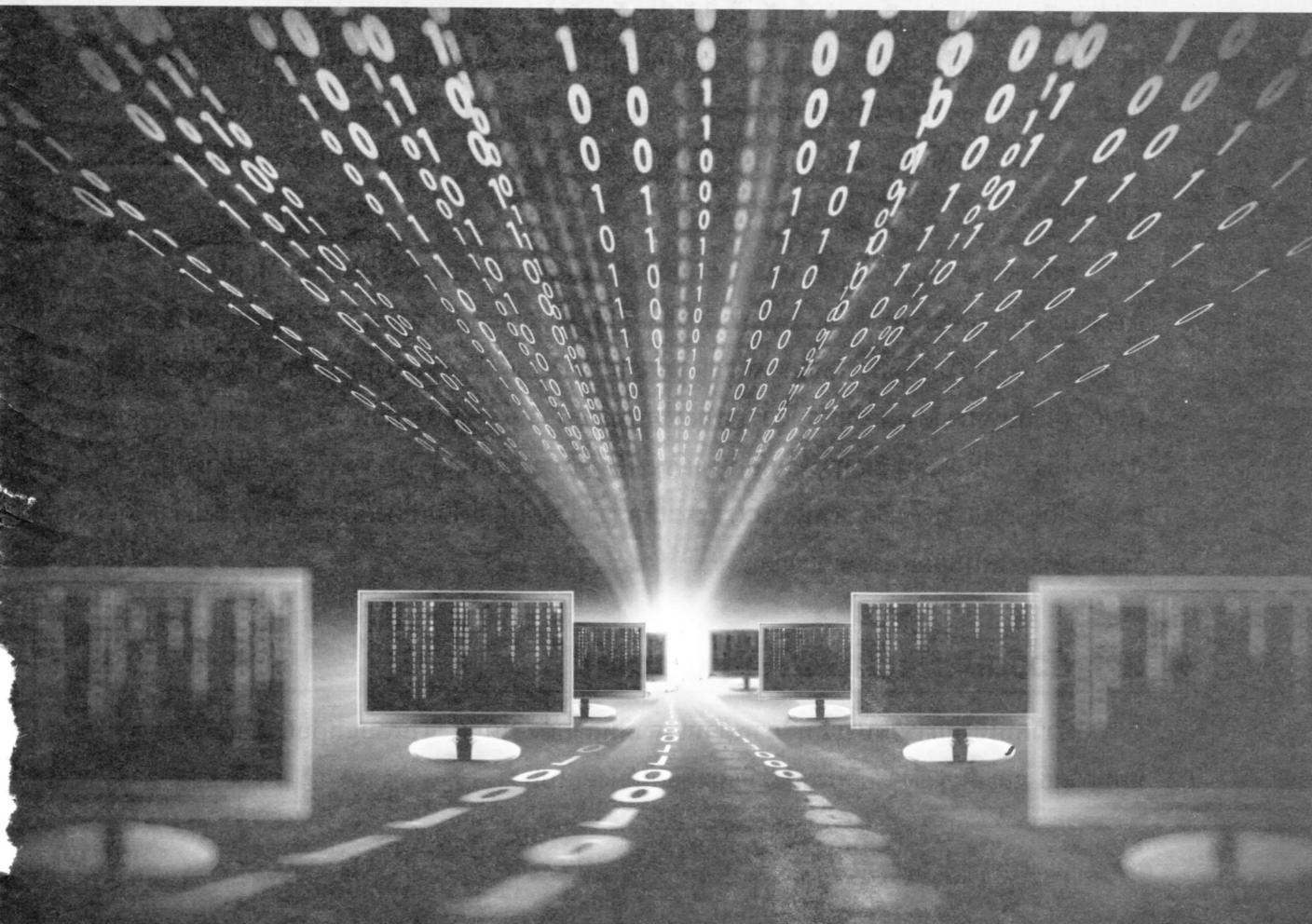


2

ตัววัดเชิงปริมาณ

(Quantitative Measure)



ในการออกแบบสถาปัตยกรรมคอมพิวเตอร์ จำเป็นจะต้องพิจารณาปัจจัยต่างๆ เพื่อแสดงให้เห็นถึงประสิทธิภาพของเครื่องคอมพิวเตอร์ ในที่นี้จะเน้นประสิทธิภาพในด้านเวลา โดยเฉพาะเวลาในการประมวลผลเป็นสำคัญ แม้ว่าจะมีประสิทธิภาพในด้านอื่นๆ ที่น่าสนใจ เช่น การใช้พลังงาน ซึ่งเป็นสิ่งที่ได้รับความสนใจในปัจจุบัน ในบทนี้จะกล่าวถึงตัววัดเชิงปริมาณแบบต่างๆ ความหมาย และตัวอย่างการใช้งาน เพื่อเป็นพื้นฐานในการวิเคราะห์ประสิทธิภาพของเครื่องคอมพิวเตอร์นั่นเอง

จากคำกล่าวของนาย Brook ที่ว่า อัตราส่วนระหว่างราคาและประสิทธิภาพของเครื่องนั้นเป็นอยู่กับการอิมพลีเมนต์รองฮาร์ดแวร์ แต่ในแง่ของการใช้งาน ทั้งในด้านการใช้และการเขียนโปรแกรมนั้น ปัจจุบันนักออกแบบสถาปัตยกรรม การสร้างเครื่องคอมพิวเตอร์ที่ประกอบด้วยเมนบอร์ด ชิปปี้ และหน่วยความจำ การเลือกใช้เทคโนโลยีสำหรับการอิมพลีเมนต์แต่ละองค์ประกอบเหล่านี้ มีผลกับเวลาในการทำงานของแต่ละหน่วยและเวลาจากการหับหั่น รวมทั้งราคาด้วย เช่น ถ้าใช้หน่วยความจำแบบ Static (SRAM) ก็จะเก็บข้อมูลได้ดีกว่าและเร็วกว่าแบบ Dynamic (DRAM) แต่ว่าจะมีราคาสูงกว่า สิ่งเหล่านี้แม้จะไม่ได้เกี่ยวข้องกับโปรแกรมเมอร์แต่อย่างไร เพราะการเข้าถึงหน่วยความจำที่ยังคงเป็นวิธีปกติตามหลักการโปรแกรมทั่วไป ในการกำหนดวิธีการเข้าถึงของโปรแกรมต่างๆ นั้น เป็นหน้าที่ของนักออกแบบสถาปัตยกรรม ซึ่งหน้าที่เกี่ยวข้องกับการออกแบบชุดคำสั่งให้โปรแกรมเมอร์เรียกใช้ได้ ต้องออกแบบให้ทำงานเร็วในระดับฮาร์ดแวร์และมีราคากลางๆ

มุ่งมองต่างๆ ในระดับผู้ใช้งานและระดับนักออกแบบสถาปัตยกรรมจะไม่เหมือนกัน ผู้ใช้งานอาจจะมองที่ประสิทธิภาพที่ดีที่สุด ในราคาน้ำที่ประหยัดที่สุด ในขณะที่นักออกแบบสถาปัตยกรรมอาจจะมองลึกถึงการปรับปรุงประสิทธิภาพด้วยค่าใช้จ่ายที่น้อยที่สุด โดยการปรับปรุงดังกล่าวอาจจะเป็นการปรับปรุงบางส่วนของฮาร์ดแวร์ เพิ่มองค์ประกอบ แก้ไของค์ประกอบ เพิ่มรูปแบบคำสั่งสำหรับใช้งาน โดยจะให้มีผลกระทบต่อโปรแกรมของผู้ใช้งานน้อยที่สุด ไม่ว่าจะเป็นกรณีใดก็ตาม จะต้องมีปัจจัยในการวัดหรือการประเมิน ดังจะได้กล่าวถึงต่อไป แต่ในเบื้องต้นจะขอกล่าวถึงการวัดประสิทธิภาพในมุมมองของเวลา ก่อน

2.1 ตัววัดประสิทธิภาพต่างๆ ด้านเวลา

ตัววัดที่ใช้ในการวัดประสิทธิภาพด้านเวลาไม่หลายตัว ได้แก่ Throughput, Latency, แบนด์วิดธ์ และอื่นๆ ซึ่งจะอธิบายถึงรายละเอียดได้ดังต่อไปนี้

- Throughput หมายถึง จำนวนงานที่ทำต่อหนึ่งหน่วยเวลา
- Latency หรือ Response Time (Execution Time) หมายถึง เวลาทั้งหมดที่ใช้ในการทำงานหนึ่งๆ โดยทั่วไปแล้ว การเพิ่ม Throughput หมายถึง การลดเวลาในการประมวลผล (Execution Time) นั่นเอง
- แบนด์วิดธ์ หมายถึง อัตราการรับส่งข้อมูลต่อหนึ่งหน่วยเวลา

ในการวัดเวลาในการทำงานของโปรแกรม (ในที่นี้เราจะเรียกเวลาในการทำงานของโปรแกรมที่ใช้พิจารณา Execution Time) โดยทั่วไปแล้ว ประสิทธิภาพจะเป็นส่วนกลับของ Execution Time ก้าวคือ $\text{Performance} = \frac{1}{\text{Execution Time}}$ ดังนั้น การเพิ่มประสิทธิภาพจะหมายถึงการลดเวลาที่โปรแกรมใช้ในการทำงานนั่นเอง

สำหรับสูตรในการคำนวณหา Execution Time หรือหา CPU Time นั้น ได้มาจาก

$$\text{CPU Time ที่ใช้ต่อโปรแกรม} = \text{จำนวนไบเกลที่ใช้ต่อโปรแกรม} \times \text{Clock Cycle Time}$$

โดยทั่วไปแล้ว เวลาในการทำงานของโปรแกรมหนึ่งๆ นั้นรัดได้จากจำนวนไบเกลที่ใช้ต่อโปรแกรม ซึ่งเป็นจำนวนรอบสัญญาณนาฬิกาที่ใช้สำหรับทำงานโปรแกรมนั้นๆ ส่วน Clock Cycle Time นั้น หมายถึง ความยาวของหนึ่งรอบสัญญาณนาฬิกาซึ่งเป็นส่วนกลับของความถี่สัญญาณนาฬิกา (Clock Rate)¹ เชียนอีกอย่างได้ว่า

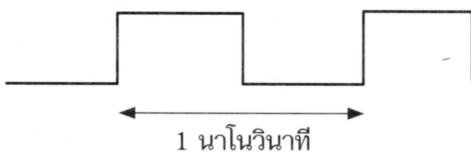
$$\text{CPU Time ที่ใช้ต่อโปรแกรม} = \frac{\text{จำนวนไบเกลที่ใช้ต่อโปรแกรม}}{\text{Clock Rate}}$$

จากสมการของ CPU Time ข้างต้น การทำให้เวลาในการทำงานเร็วขึ้นจะต้องลดจำนวนไบเกลที่ใช้ในการทำงานของโปรแกรมนั้น หรือเพิ่ม Clock Rate นั่นเอง

¹ เช่น เครื่องคอมพิวเตอร์ที่มีความถี่ 500 เมกะเฮิรตซ์ ก็จะมี Clock Cycle Time เป็น

$$\frac{1}{500 \times 10^6} = \frac{1}{500,000,000} = 0.5 \times 10^{-9} = 0.5 \text{ นาโนวินาที}$$

ลองมาพิจารณาความหมายของ Clock Cycle (เรียกสั้นๆ ว่าไซเกล) ซึ่งเป็นเวลาที่ใช้ใน 1 รอบของสัญญาณนาฬิกา เช่น ถ้าเครื่องคอมพิวเตอร์มีความเร็ว (หรือความถี่) เท่ากับ 1 กิกะเฮิรตซ์ ใน 1 รอบของสัญญาณนาฬิกาจะยาวเท่ากับ $1/(1,000 \times 10^6)$ หรือ 1 นาโนวินาที นั่นเอง (1 นาโนวินาที = 10^{-9} วินาที) ดังแสดงในรูปที่ 2.1



รูปที่ 2.1 รอบสัญญาณนาฬิกาสำหรับสัญญาณนาฬิกา (Clock Rate) ที่ความถี่ 1 กิกะเฮิรตซ์

เราเรียก 1 กิกะเฮิรตซ์ นี้ว่า Clock Rate ดังนั้น ในสมการข้างต้น Clock Rate จะเป็นส่วนกลับของ Clock Cycle หากกล่าวว่าเครื่องคอมพิวเตอร์หนึ่งมีความถี่สัญญาณนาฬิกาเท่ากับ 200 เมกะเฮิรตซ์ ก็หมายถึงว่ามี Clock Cycle Time (ค่า หรือ Period) เท่ากับ $1/(200 \times 10^6) = 5$ นาโนวินาทีนั่นเอง แต่ว่าใน 1 รอบสัญญาณนาฬิกานี้อาจจะทำงานได้ 1 คำสั่ง หรืออาจจะยาวไม่พอที่จะทำงาน 1 คำสั่ง (ดังจะได้อธิบายถึงต่อไป) นั่นคือ แท้จริงแล้วมีบางคำสั่งอาจจะใช้เวลามากกว่า 1 ไซเกลก็ได้ ซึ่งอาจจะเป็นคำสั่งที่ซับซ้อน เช่น คำสั่งเกี่ยวกับการคูณหรือการหาร คำสั่งเกี่ยวกับเลขทศนิยม คำสั่งเกี่ยวกับการเข้าถึงหน่วยความจำ เป็นต้น

พิจารณาตัวอย่างการคำนวณง่ายๆ ต่อไปนี้

ตัวอย่างที่ 2.1 ถ้าเรามีโปรแกรมหนึ่งทำงานเสร็จในเวลา 10 วินาที เมื่อทำงานบนเครื่องคอมพิวเตอร์ A ซึ่งมีความถี่ 400 เมกะเฮิรตซ์ เราต้องการจะออกแบบเครื่องคอมพิวเตอร์ใหม่ เรียกว่าเครื่องคอมพิวเตอร์ B ซึ่งจะทำงานกับโปรแกรมนี้ในเวลา 7 วินาที นักออกแบบอาจจะใช้เทคโนโลยีใหม่ที่จะทำให้สัญญาณนาฬิกาเร็วขึ้น แต่อาจจะมีผลกระทบต่อส่วนอื่นๆ ของชีพิญ โดยจะทำให้เครื่อง B นี้ใช้จำนวนไซเกลมากกว่าเครื่อง A อยู่ 1.2 เท่าสำหรับการทำงานโปรแกรมเดียวกันนี้ ดังนั้น เราควรจะออกแบบให้เครื่องคอมพิวเตอร์ B มี Clock Rate เท่าไรจึงจะทำงานโปรแกรมนี้เสร็จภายใน 7 วินาที

วิธีทำ

เครื่องคอมพิวเตอร์ A มีความถี่ 400 เมกะเฮิรตซ์ หมายถึงมี Cycle Time = $1/(400 \times 10^6) = 0.25 \times 10^{-8}$ วินาที

โปรแกรม A ทำงานเสร็จใน 10 วินาที ดังนั้น จะทำงานโดยใช้จำนวนไชเกิลเท่ากับ $10/(2.5 \times 10^{-9}) = 4 \times 10^9$ ไชเกิล

และทำงานบนเครื่องคอมพิวเตอร์ B ใช้สัญญาณนาฬิกาที่เร็วมากกว่าเดิม 1.2 เท่า คือใช้จำนวนไชเกิลเท่ากับ $1.2 \times 4 \times 10^9 = 4.8 \times 10^9$ ไชเกิล

ถ้าต้องการทำงานให้เสร็จภายใน 7 วินาที จะใช้จำนวนไชเกิลเท่ากับ $4.8/(7 \times 10^9) = 0.7 \times 10^9$ ลูก

นั่นคือเครื่อง B ต้องมีความถี่ 700 เมกะเฮิรตซ์

ลองพิจารณาตัววัดอื่นๆ บ้าง เช่น CPI (Cycle Per Instruction) ซึ่งหมายถึงจำนวนไชเกิลที่ใช้ต่อการทำงาน 1 คำสั่ง หรืออาจจะหมายถึงจำนวนไชเกิลโดยเฉลี่ยที่ใช้ต่อการทำงาน 1 คำสั่ง การหา CPI ที่แท้จริง (Effective CPI) โดยทั่วไปหาได้จากสูตร

$$\text{Effective CPI} = \sum_{i=1}^n (\text{CPI}_i \times \text{IC}_i)$$

โดยที่ CPI_i คือ จำนวนไชเกิลที่ใช้ทำงาน โดยคำสั่งแต่ละประเภทอาจ ได้แก่ ประเภท Load/Store คำสั่งประเภทการคำนวน เป็นต้น การแบ่งประเภทคำสั่งจะได้กล่าวถึงในบทต่อไป

IC_i คือ จำนวนคำสั่งในโปรแกรมที่เป็นคำสั่งประเภท i โดยย่อมาจาก Instruction Count และ n คือ จำนวนประเภทคำสั่งทั้งหมดในชุดคำสั่นนนๆ

การหา Effective CPI นี้จะต้องทดสอบกับหลายๆ โปรแกรม หรือที่เรียกว่าชุด Benchmark (ดังจะได้กล่าวถึงต่อไป) ที่มีลักษณะรูปแบบโปรแกรมและจำนวนคำสั่งต่างกัน

ในการออกแบบชุดคำสั่งแท้จริงแล้วมีผลต่อ CPI เฉลี่ยนี้ เพราะจะเห็นว่าคำสั่งแต่ละคำสั่งอาจจะใช้จำนวนไชเกิลไม่เท่ากันได้

ตัวอย่างที่ 2.2 พิจารณาโปรแกรมหนึ่งๆ ซึ่งทำงานบนเครื่องต่างกันดังนี้

สำหรับเครื่องคอมพิวเตอร์ A มี Clock Cycle Time เท่ากับ 10 นาโนวินาที และมี CPI = 2.0

สำหรับเครื่องคอมพิวเตอร์ B มี Clock Cycle Time เท่ากับ 20 นาโนวินาที และมี CPI = 1.3

โปรแกรมนี้ทำงานบนเครื่องคอมพิวเตอร์ใดเร็วกว่ากัน และเร็วกว่าเท่าไร

วิธีทำ

เครื่องคอมพิวเตอร์ A ทำงาน 1 คำสั่งใช้เวลา 20 นาโนวินาที เครื่องคอมพิวเตอร์ B ทำงาน 1 คำสั่งใช้เวลา 26 นาโนวินาที

ดังนั้น เครื่องคอมพิวเตอร์ A เร็วกว่าเครื่องคอมพิวเตอร์ B อยู่ 1.3 เท่า

พิจารณากรณีที่นักสถาปัตยกรรมพิจารณาตัวแปลภาษาหรือคอมไพล์เยอร์ (Compiler) สำหรับชุดคำสั่งและสถาปัตยกรรมของตน โดยต้องการเลือกวิธีอوبติไมโครโค้ดจากสองรูปแบบ สำหรับเครื่องคอมพิวเตอร์ X สำหรับเครื่องนี้มีรูปแบบคำสั่งที่ใช้ได้ 3 แบบ ได้แก่ แบบ A แบบ B และแบบ C ซึ่งแต่ละแบบใช้งานนี้เกลเท่ากับ 1, 2 และ 3 ตามลำดับ

พิจารณากรณีของโค้ดที่ถูกสร้างโดยตัวแปลภาษานี้ด้วยการอوبติไมโครโค้ดต่างๆ กัน ดังนี้

กรณีที่ 1 โค้ดที่ได้จากการอوبติไมโครโค้ดด้วย คำสั่งรูปแบบ A จำนวน 2 คำสั่ง คำสั่งรูปแบบ B จำนวน 1 คำสั่ง คำสั่งรูปแบบ C จำนวน 2 คำสั่ง

กรณีที่ 2 โค้ดที่ได้จากการอوبติไมโครโค้ดด้วยหั้งหมด 6 คำสั่ง มีคำสั่งรูปแบบ A จำนวน 4 คำสั่ง คำสั่งรูปแบบ B จำนวน 1 คำสั่ง คำสั่งรูปแบบ C จำนวน 2 คำสั่ง

อยากรู้ว่าโค้ดกรณีใดจะทำงานได้เร็วกว่า และเร็วกว่าอยู่เท่าไร

ให้หา CPI ของโค้ดแต่ละรูปแบบด้วย

แบบที่ 1 ใช้ $2 + 2 + 6 = 10$ ไซเกิล

แบบที่ 2 ใช้ $4 + 2 + 6 = 12$ ไซเกิล

$$\text{ตั้งนั้น } CPI_1 = 10/5 = 2 \text{ และ}$$

$$CPI_2 = 12/7 = 1.7$$

ตั้งนั้น ถ้าใช้ CPI เป็นตัววัด จะพบว่าโค้ดกรณีที่ 2 ดีกว่า และดีกว่าอยู่ $2/1.7 = 1.18$ เท่า

การใช้ MIPS (Million Instruction Per Second) ก็เป็นอีกทางเลือกในการวัดประสิทธิภาพ โดย MIPS โดยทั่วไปแล้วจะหมายถึงจำนวนคำสั่งที่ทำงานหนึ่งล้านในหน่วยวินาที ลองพิจารณาตัวอย่างต่อไปนี้

ตัวอย่างที่ 2.3 สำหรับชุดคำสั่งที่ประกอบด้วย 3 รูปแบบ ได้แก่ แบบ A แบบ B และแบบ C ซึ่งแต่ละรูปแบบใช้เวลาในการทำงานเท่ากัน 1 ไบเกิล 2 ไบเกิล และ 3 ไบเกิล ตามลำดับ พิจารณาตัวแปรภาษาที่สามารถสร้างโค้ดออกมา 2 รูปแบบ คือ

รูปแบบที่ 1 โค้ดประกอบด้วยคำสั่งแบบ A จำนวน 5 ล้านคำสั่ง คำสั่งแบบ B จำนวน 1 ล้านคำสั่ง และคำสั่งแบบ C จำนวน 1 ล้านคำสั่ง

รูปแบบที่ 2 โค้ดประกอบด้วยคำสั่งแบบ A จำนวน 10 ล้านคำสั่ง คำสั่งแบบ B จำนวน 1 ล้านคำสั่ง และคำสั่งแบบ C จำนวน 1.5 ล้านคำสั่ง

จงหาว่าโค้ดรูปแบบใดจะทำงานเร็วกว่าหากใช้ MIPS เป็นตัววัด และรูปแบบใดจะทำงานเร็วกว่าถ้าใช้ Execution Time เป็นตัววัด

วิธีทำ

สำหรับเครื่องคอมพิวเตอร์ความถี่ 200 เมกะเฮิรตซ์ หมายถึงว่า Clock หนึ่งลูกใช้เวลา $1/(200 \times 10^6) = 1/(2 \times 10^8)$ วินาที

แบบที่ 1 ใช้เวลา $5 + 2 + 3 = 10$ ล้านไบเกิล

แบบที่ 2 ใช้เวลา $10 + 2 + 4.5 = 16.5$ ล้านไบเกิล

แบบที่ 1 ใช้เวลา $10 \times 10^6 \times 1/(2 \times 10^8) = 5 \times 10^{-2}$ วินาที

แบบที่ 2 ใช้เวลา $16.5 \times 10^6 \times 1/(2 \times 10^8) = 6.25 \times 10^{-2}$ วินาที

$$\text{MIPS1} \text{ เท่ากับ } 6 \text{ ล้าน}/(5 \times 10^{-2}) = 1.2 \times 10^2 = 120$$

$$\text{MIPS2} \text{ เท่ากับ } 16.5 \text{ ล้าน}/(6.25 \times 10^{-2}) = 2.64 \times 10^2 = 264$$

ดังนั้นถ้าใช้ MIPS เป็นตัววัด จะเห็นว่าแบบที่ 2 จะดีกว่า เพราะให้ค่า MIPS มากกว่า กล่าวคือ แสดงว่าเครื่องทำงานได้เร็ว แต่ถ้าใช้ Execution Time เป็นตัววัด แบบที่ 1 จะดีกว่า เพราะว่าใช้เวลาในการทำงานน้อย

โดยทั่วไปแล้ว การวัดประสิทธิภาพนั้นจะทำได้ดีเมื่อเราทดสอบจริงบนเครื่องคอมพิวเตอร์ที่โปรแกรมนั้นๆ ทำงาน โดยอาจจะมองถึงตัวอย่างโปรแกรมที่จะใช้ และสามารถแสดงให้เห็นถึงโหลดงานของเครื่องจริงๆ ได้ หรืออาจจะเป็นตัวอย่างโปรแกรมที่ใช้เครื่องคอมพิวเตอร์ทั่วไปมากจะใช้งาน เช่น ตัวแปลภาษา Editor โปรแกรมด้านวิทยาศาสตร์ โปรแกรมด้านกราฟิก เป็นต้น โดยโปรแกรมเหล่านี้ที่ใช้ในการทดสอบประสิทธิภาพที่เรียกว่าชุด Benchmark ประกอบด้วยโปรแกรมหลายๆ ตัวจัดรวมกันเป็นชุด ถูกสร้างขึ้นมาสำหรับการทดสอบเครื่องให้เป็นไปตามมาตรฐาน เพื่อให้เห็นประสิทธิภาพของเครื่องนั้นๆ ในด้านต่างๆ กันเทียบกับเครื่องอื่นๆ ได้ ชุด Benchmark มาตรฐานนั้นเรียกว่า SPEC (www.spec.org) ซึ่งมีหลายรุ่น แต่ละรุ่นอาจจะเหมาะสมกับการทดสอบที่ต่างกัน หมายกันก็ได้ ตัวอย่างข้างล่างเป็นโปรแกรมต่างๆ ที่ SPEC Benchmark บรรจุอยู่ Benchmark ชุดนี้ประกอบด้วยโปรแกรมที่ใช้การคำนวณข้อมูลประเภทจำานวนเต็ม และการคำนวณข้อมูลประเภทศักยิม ทำให้ชุดทดสอบนี้สามารถทดสอบประสิทธิภาพเครื่องได้ทั้งสองด้าน

ชุด Benchmarks สำหรับการคำนวณแบบจำนวนเต็ม		ชุด Benchmarks สำหรับการคำนวณแบบศักยิม	
gzip	โปรแกรมบีบอัดข้อมูล	wup-wise	โปรแกรม Quantum Chromodynamics
vpr	โปรแกรมเกี่ยวกับการ Route เส้นทางใน FPGA	swim	ตัวแบบ Shallow Water
gcc	ตัวแปลภาษา GNU C	mgrid	โปรแกรม Multigrid Solver แบบ 3D

(ต่อ)

ชุด Benchmarks สำหรับการคำนวณแบบจำนวนเต็ม		ชุด Benchmarks สำหรับการคำนวณแบบทศนิยม	
mcf	โปรแกรมอปติไมซ์คอมไบนา-	applu	โปรแกรม Parabolic/Elliptic pde
thorik	ทอริก		
crafty	โปรแกร姆 Chess	mesa	โปรแกร姆 3D Graphics Library
parser	โปรแกร姆 Word Processing	galgel	โปรแกรม Computational Fluid Dynamics
eon	Computer Visualization	art	โปรแกรม Image Recognition (NN)
perlbench	โปรแกรม Perl	equake	การจำลอง Seismic Wave Propagation
gap	Interpreter ของ Group Theory	facerec	โปรแกรม Facial Image Recognition
vortex	ฐานข้อมูลเชิงวัตถุ	ammp	โปรแกรม Computational Chemistry
bzip2	โปรแกรมบีบอัดข้อมูล	lucas	การทดสอบจำนวนเฉพาะ
twolf	โปรแกรมวางแผนจรา	fma3d	โปรแกรม Crash Simulation fem
		sixtrack	โปรแกรม Nuclear Physics
		apsi	โปรแกรม Pollutant Distribution

ที่มา: <http://www.spec.org>

■ 2.2 กำลังและพลังงาน

ตัววัดประสิทธิภาพด้านอื่นๆ ที่สำคัญ ได้แก่ การใช้กำลัง (Power) และพลังงาน (Energy) ซึ่งทั้ง 2 ตัวนี้มีความสำคัญกับระบบและคอมพิวเตอร์แบบฝังตัว เนื่องจากระบบฝังตัวต้องการประหยัดการใช้แบตเตอรี่ให้มากที่สุด และลดความร้อนที่เกิดขึ้นระหว่างการประมวลผล

จากสมการของการทำกำลังที่ใช้กัน ได้แก่

$$\text{Power}_{\text{dynamic}} = \frac{1}{2} \times \text{CapacitiveLoad} \times \text{Voltage}^2 \times \text{FrequencySwitched}$$

สูตรนี้จะคำนวณกำลังที่เกิดจากการใช้งาน โดยพิจารณาจากโหลดของตัวเก็บประจุ โวลเตจ (Voltage) และความถี่ในการปิด/เปิดสวิตช์

นอกจากนี้ในบางกรณี เราอาจจะสนใจพลังงานที่เกิดขึ้นระหว่างการทำงาน คำนวณได้จาก

$$\text{Energy}_{\text{dynamic}} = \text{CapacitiveLoad} \times \text{Voltage}^2$$

โดยทั่วไปแล้ว การใช้สัญญาณนาฬิกาที่มีความถี่น้อย หมายถึง จะมีการสลับสวิตช์ที่น้อย (Frequency Switch) จะเป็นการลดการใช้กำลัง ไม่ใช่การลดการใช้พลังงาน สำหรับตัวแปร Capacitive Load เป็นตัวแปรที่เกี่ยวข้องกับจำนวนทรานซิสเตอร์ในชาร์ดแวร์นั้นๆ ซึ่งหมายถึงการเก็บประจุหรือค่าพาหะแทนของทรานซิสเตอร์ที่ใช้ทั้งหมด รวมไปถึงการเดินสาย (Wiring) ต่างๆ ภายในชาร์ดแวร์ด้วย จะเห็นว่าการลดโวลเตจ (Voltage) เช่น เปลี่ยนจากการใช้ระบบไฟ 5 โวลต์ เป็น 3 โวลต์ ก็จะลดการใช้กำลังด้วยเช่นกัน นอกจากนี้ การลดการใช้งานหน่วยประมวลผล เช่น การใช้ Sleep Mode หรือ Standby Mode ก็จะเป็นการลดการใช้กำลัง เช่น อาจจะปิดสัญญาณนาฬิกาที่เกี่ยวกับหน่วยทศนิยมถ้าไม่จำเป็นต้องใช้หน่วยนี้ เป็นต้น

ตัวอย่างเช่น ถ้าต้องการลดโวลเตจ 17% และการลดโวลเตจนี้จะทำให้ลดการสลับสวิตช์ ให้น้อยลง 17% ด้วย ดังนั้นจะลด Dynamic Power ไปอย่างไร

$$\begin{aligned}\text{Power}_{\text{dynamic}} &= \frac{1}{2} \times \text{CapacitiveLoad} \times \text{Voltage}^2 \times \text{FrequencySwitched} \\ &= \frac{1}{2} \times 0.83 \times \text{CapacitiveLoad} \times (0.83 \times \text{Voltage})^2 \\ &\quad \times \text{FrequencySwitched} \\ &= (0.83)^3 \times \text{OldPower}_{\text{dynamic}} \\ &\approx 0.6 \times \text{OldPower}_{\text{dynamic}}\end{aligned}$$

โดยปกติแล้ว แม้จะมีการปิดเครื่องก็ยังมีกำลังที่ต้องการใช้งานอยู่ กำลังนี้เรียกว่า Static Power

$$\text{Power}_{\text{static}} = \text{Current}_{\text{static}} \times \text{Voltage}$$

กระแสไฟฟ้า (Current) นี้เรียกว่า Leakage Current ซึ่งเป็นไฟเลี้ยงที่ต้องใช้สำหรับทรานซิสเตอร์แต่ละตัว หากชาร์ดแวร์มีจำนวนทรานซิสเตอร์มากจะมี Leakage Current จำนวนมากตามด้วย ดังนั้น จะเห็นว่าการเพิ่มทรานซิสเตอร์ไปในชาร์ดแวร์ก็จะเป็นการเพิ่มการใช้ Power เช่น

กัน ในปี ค.ศ. 2006 ได้มีผู้ทำนายว่าในชาร์ดแวร์หนึ่งๆ จะมี Leakage Current ประมาณ 25% หรือมากสุดถึง 40%

■ 2.3 การวัดประสิทธิภาพและปัจจัยที่เกี่ยวข้อง

จากปัจจัยต่างๆ ข้างต้น สรุปได้ว่าสามารถประลิทธิภาพด้านความเร็วนั้นเกิดจากความเร็วในการประมวลผล ดังสมการ

$$\text{CPU Time} = \text{Instruction, Count} \times \text{CPI} \times \text{Clock, Cycle}$$

หรือ

$$\text{CPU Time} = \text{Instruction, Count} \times \text{CPI} \times \text{Clock, Cycle/Clock Rate}$$

ในสมการข้างต้นของการวัดประสิทธิภาพนั้น ประกอบด้วยตัวแปรหลายตัวได้แก่ Instruction Count, CPI และ Clock Rate

ลองพิจารณาการปรับเปลี่ยนค่าของตัวแปรเหล่านี้เพื่อให้ CPU Time มีค่าน้อยๆ หรือใช้เวลาประมวลผลน้อยนั่นเอง

ตัวแปร Instruction Count เป็นจำนวนคำสั่ง ซึ่งหมายถึงจำนวนคำสั่งทั้งหมดที่อยู่ภายในโปรแกรมหนึ่ง ซึ่งจะได้มาจากการที่ตัวแปลภาษาแปลงโปรแกรมที่เขียนในภาษาขั้นสูงเป็นภาษาแอสเซมบลีหรือเป็นภาษาเครื่อง และนับคำสั่งที่ใช้ในการทำงานในระดับภาษาเครื่อง การใช้วิธีการออบต์ไมโครโคดแบบต่างๆ จะทำให้จำนวนคำสั่งและรูปแบบคำสั่งที่ใช้เปลี่ยนแปลง ซึ่งจะมีผลโดยตรงกับ Instruction Count (IC) ดังนั้น การลดจำนวนคำสั่งที่ใช้ทั้งหมดจะมีส่วนในการลดเวลาโปรแกรมนั้นทำงานทั้งหมดด้วย กล่าวคือ ตัวแปร IC นั้นขึ้นตรงกับตัวแปลงภาษาที่นั่นเอง นอกจากนั้นยังขึ้นอยู่กับผู้เขียนโปรแกรมว่าใช้อัลกอริธึมอย่างไรในการเขียน และขึ้นอยู่กับภาษาโปรแกรมที่ใช้ หรือกล่าวรวมๆ คือเป็นผลจากตัวซอฟต์แวร์นั่นเอง

สำหรับตัวแปร CPI นั้น จะขึ้นอยู่กับสถาปัตยกรรมของเครื่องนั้นว่าถูกออกแบบด้วย ISA (Instruction Set Architecture) รูปแบบใด มีคำสั่งกี่รูปแบบ แต่ละรูปแบบใช้จำนวนไบต์เกิลในการทำงานเท่าไร ซึ่งเกี่ยวข้องทั้งด้านการออกแบบ ISA และโครงสร้างทางชาร์ดแวร์ของเครื่อง

สำหรับตัวแปร Clock Rate นั้นเกี่ยวข้องโดยตรงกับเทคโนโลยีทางชาร์ดแวร์ของเครื่องยุคนั้นๆ ว่าสามารถทำให้ความถี่มากสุดเป็นเท่าไร

ลองพิจารณาตัวอย่างต่อไปนี้

ถ้าโปรแกรมหนึ่งๆ ที่ใช้ทดสอบมีลักษณะที่ประกอบด้วยคำสั่งต่างๆ รูปแบบดังนี้

ประเภทคำสั่ง	ความถี่	CPI _i
ALU	50%	1
Load	20%	4
Store	10%	3
Branch	20%	2

ลองพิจารณากรณีต่อไปนี้

- เครื่องจะเร็วขึ้นกี่เท่าถ้าเราใช้แคชช่วยเก็บข้อมูลและทำให้ลดเวลาในการโหลดข้อมูลเหลือ 2 ไซเกิล
- ถ้าเบรี่ยนเทียบกับการใช้วิธีการทำนาย Branch² เพื่อลดเวลาเกี่ยวกับคำสั่ง Branch ไป 1 ไซเกิล วิธีใดจะดีกว่ากัน
- ถ้าเครื่องสามารถทำคำสั่งเกี่ยวกับ ALU ได้ 2 คำสั่งพร้อมกัน เครื่องจะเร็วขึ้นหรือไม่ ลองคำนวณว่าในส่วนของโปรแกรมนี้จะใช้เวลา กับคำสั่งแต่ละประเภทอย่างไร

ประเภทคำสั่ง	ความถี่ × CPI _{เดิม}	กรณีที่ 1 :	กรณีที่ 2 :	กรณีที่ 3 :
		ความถี่ × CPI	ความถี่ × CPI	ความถี่ × CPI
ALU	0.5	0.5	0.5	0.25
Load	0.8	0.4	1.0	1.0
Store	0.3	0.3	0.3	0.3
Branch	0.4	0.4	0.2	0.4
CPI รวม	2.0	1.6	2.0	1.95

² เป็นวิธีการคาดคะเนว่าที่พิจารณาจะกระโดดไปทำงาน ณ ตำแหน่งใด

สำหรับกรณีแรก จะเห็นว่า CPU Time ค่าใหม่ที่ได้จะเท่ากับ $1.6 \times IC \times CC$ ซึ่งเท่ากับ $2.0/1.6$ คือเร็วขึ้น 12.5%

สำหรับกรณีสอง จะเห็นว่า CPU Time ค่าใหม่ที่ได้จะเท่ากับ $2.0 \times IC \times CC$ ซึ่งเท่ากับ $2.0/2.0$ คือไม่เร็วขึ้น

สำหรับกรณีสาม จะเห็นว่า CPU Time ค่าใหม่ที่ได้จะเท่ากับ $1.95 \times IC \times CC$ ซึ่งเท่ากับ $2.0/1.95$ คือเร็วขึ้น 10.3%

ดังนั้น จะสรุปได้ว่าวิธีการแรกจะให้ผลต่ำกว่าวิธีการอื่นๆ เพราะจะเร็วที่สุด

ในการนี้ที่เราทดสอบกับ Benchmark ซึ่งประกอบด้วยหลายๆ โปรแกรม เราอาจจะต้องหาวิธีสรุปค่า CPU Time ที่ได้จากแต่ละโปรแกรมก่อนจะมาพิจารณาหา CPU Time จากสูตรนี้ วิธีการสรุปค่า CPU Time อาจจะใช้สูตรต่อไปนี้

$$AM = \frac{1}{n} \sum Time_i$$

ซึ่งเป็นการหาค่าเฉลี่ยเลขคณิต (Arithmetic Mean) นั่นเอง อาจจะลองพิจารณาการหาค่าเฉลี่ยแบบอื่นๆ ก็ได้ เช่น ค่าเฉลี่ยเรขาคณิต (Geometric Mean) เป็นต้น

$$\text{GeometricMean} = \sqrt[n]{\prod_{i=1}^n \text{SPECRatio}_i}$$

โดยที่ SPECRatio คือค่า Execution Time ของ Benchmark ที่ทำงานได้บนเครื่องที่ต้องการวัดประสิทธิภาพเทียบกับเครื่องคอมพิวเตอร์อ้างอิง โดยสามารถหาจาก

$$\text{Execution Time บนเครื่องคอมพิวเตอร์ X} / \text{Execution Time บนเครื่องทดสอบ}$$

ถ้า SPECRatio ของเครื่องคอมพิวเตอร์ A มากกว่าของเครื่องคอมพิวเตอร์ B อยู่ 1.25 เท่า หมายถึง

$$1.25 = \frac{\text{SPECRatio}_A}{\text{SPECRatio}_B} = \frac{\frac{\text{ExecutionTime}_{\text{reference}}}{\text{ExecutionTime}_A}}{\frac{\text{ExecutionTime}_B}{\text{ExecutionTime}_B}}$$

$$= \frac{\text{ExecutionTime}_B}{\text{ExecutionTime}_A} = \frac{\text{Performance}_A}{\text{Performance}_B}$$

อาจจะมีการคำนวณหาล็อกฟังก์ชันของ SPECRatio หาก่าเฉลี่ย และหา Standard Deviation เพื่อนอกพฤติกรรมหรือลักษณะของโปรแกรมในชุด Benchmark ได้ว่า

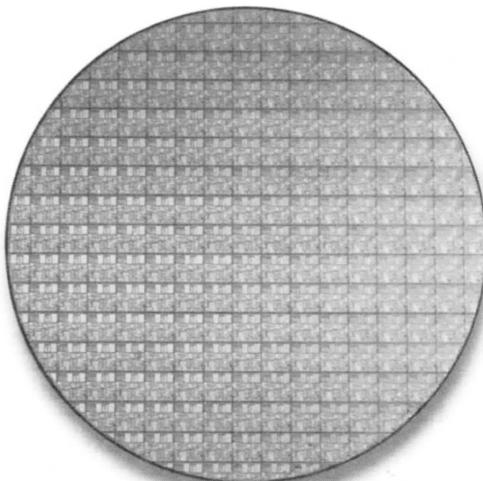
$$\text{GeometricMean} = \exp\left(\frac{1}{2} \times \sum_{i=1}^n \ln(\text{SPECRatio}_i)\right)$$

$$\text{GeometricMean} = \exp(\text{StDev}(\ln(\text{SPECRatio}_i)))$$

■ 2.4 ต้นทุนและปัจจัยที่เกี่ยวข้อง

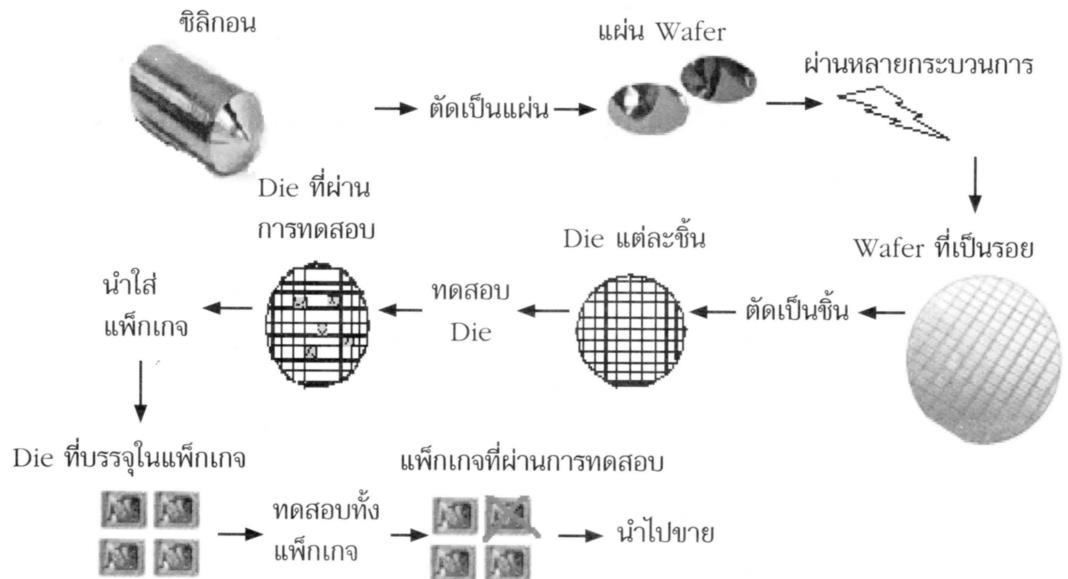
ในการผลิตชิปของชีพียู มีปัจจัยที่เกี่ยวข้องกับต้นทุน (Cost) หลายอย่าง เช่น เวลา ความนิยมของสินค้า ปริมาณ และการบรรจุสินค้า (Packaging)

หากปริมาณการผลิตมาก จะทำให้ต้นทุนต่ำ ถ้าสินค้านั้นมีความนิยมมาก มีความแพร่หลายมาก มีหลายผู้ผลิต ก็ทำให้ต้นทุนถูก เช่นกัน ถ้าสินค้าถูกห่อหุ้มสวยงาม ก็อาจจะเพิ่มต้นทุนต่อหน่วยได้ ใน การผลิตชิปหนึ่งตัว จะทำที่ลงมากๆ โดยจะทำเป็นรูป Wafer ดังแสดงในรูปที่ 2.2 และรูปที่ 2.3 แสดงขั้นตอนทั้งหมด ซึ่งเริ่มจากการใช้ชิลิกอนมาผลิตเป็น Wafer ซึ่งต้องผ่านหลายกระบวนการ จากนั้นจะทำการแยกออกมาเป็น Die และทดสอบ Die แต่ละตัวที่ได้ จากนั้นจะทำการนำไปบรรจุ เป็นแพ็คเกจออกมาเป็นชิปที่เราใช้งาน ดังแสดงตัวอย่างบอร์ดในรูปที่ 2.4

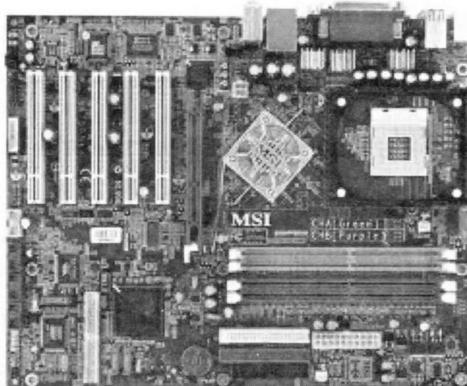


รูปที่ 2.2 Die Wafer

(ที่มา: http://www.tweaktown.com/news/12568/tsmc_shows_off_28nm_wafer_promises_it_by_2010/index.html)



รูปที่ 2.3 ขั้นตอนการผลิตชิป



รูปที่ 2.4 บอร์ดที่บรรจุชิปแล้ว

(ที่มา: <http://www.ixbt.com/mainboard/images/i875p-roundup/875pneo-fis2r-msi-board.jpg>)

ในการคำนวณต้นทุนพื้นฐาน จะทำได้ดังนี้

- ต้นทุนของ Die = ต้นทุนของ Wafer/(จำนวน Die ใน Wafer × Die Yield) กล่าวคือ ต้นทุนของ Die จะเท่ากับต้นทุนของ Wafer หารด้วยจำนวน Die ต่อ Wafer คูณด้วยสัดส่วนของ Die ที่ใช้ได้ (Die Yield)

$$\text{จำนวน Die ต่อ Wafer} = \frac{\pi \times (\text{เส้นผ่านศูนย์กลางของ Wafer}/2)^2}{\text{พื้นที่ Die}}$$

$$= \frac{\pi \times \text{เส้นผ่านศูนย์กลางของ Wafer}}{\sqrt{\text{พื้นที่ Die}}}$$

ตัวอย่างเช่น สมมติให้ Wafer มีเส้นผ่านศูนย์กลางขนาด 20 เซนติเมตร และ Die มีขนาดกว้างยาว = 1.5 เซนติเมตร ดังนั้น พื้นที่ของ Die = $1.5 \times 1.5 = 2.25$ ตารางเซนติเมตร และจะได้จำนวน Die ต่อเวเฟอร์เท่ากับ

$$\frac{\pi \times (20/2)^2}{2.25} - \frac{\pi \times 20}{\sqrt{2 \times 2.25}} = 110$$

สำหรับการตั้งราคาขาย โดยทั่วไปจะประกอบด้วย

- ต้นทุนของสินค้าแต่ละชิ้น
- ต้นทุนของสิ่งที่เกี่ยวข้องกับการผลิตสินค้าโดยตรง โดยมากจะประมาณ 10% ถึง 30%
- ต้นทุนอื่นๆ ที่ผู้ขายเพิ่มเข้าไป (Gross Margin) เกิดจากค่าใช้จ่ายอื่นๆ เช่น การโฆษณา การรักษา การทำริจัย เป็นต้น เมื่อรวมกับข้อที่ 1 และข้อที่ 2 จะได้ราคาขายโดยเฉลี่ย และ Gross Margin จะประมาณ 10% ถึง 45% ของราคาขายเฉลี่ย
- จำนวนนำมารวมกับส่วนลดที่บริษัทจะให้ได้มากที่สุด ก็จะกลายเป็น List Price

ดังตัวอย่างในรูปที่ 2.5 ที่นำต้นทุนของวัสดุมาบวกเพิ่ม 20% สำหรับค่าใช้จ่ายอื่นๆ จากนั้นมาบวก Gross Margin อีก 33% ของต้นทุนทั้งหมด และนำไปบวกด้วยส่วนลดที่เป็นไปได้อีก 33% ของราคาน้ำหนักจาก Gross Margin และให้เป็นราคาตั้งสำหรับการขาย

List Price



รูปที่ 2.5 ตัวอย่างการตั้งราคาขาย

■ 2.5 สรุป

การกล่าวว่าคอมพิวเตอร์เครื่องหนึ่งเร็วกว่าอีกเครื่องหนึ่งนั้น จะต้องมีตัววัดเชิงปริมาณที่เหมาะสม โดยทั่วไปแล้วเราจะดูความเร็วในการทำงานของโปรแกรม โดยโปรแกรมที่นำมาทดสอบนั้นควรจะมีมาตรฐานจึงจะทำให้คำล่าว่าเครื่องคอมพิวเตอร์ A เร็วกว่าเครื่องคอมพิวเตอร์ B เชื่อถือได้ ในบทนี้ได้กล่าวถึงตัววัดด้านเวลาหลายๆ ตัว เช่น CPU Time, ประสิทธิภาพ (Performance), MIPS, Throughput, Latency เป็นต้น ซึ่งตัววัดเหล่านี้เป็นตัวแสดงความเร็วของเครื่องทั้งสิ้น แต่ละตัววัดนั้นมีจุดมุ่งหมายในการวัดที่ต่างกัน นอกจากนี้ยังมีตัววัดด้านกำลังและพลังงานซึ่งเป็นสิ่งที่น่าสนใจในปัจจุบัน เพราะจะแสดงถึงการใช้พลังงานของเครื่องหนึ่งๆ ว่าต้องการพลังงานมากน้อยเพียงไรในการประมวลผล นอกจากนี้ในบทนี้ยังกล่าวถึงปัจจัยด้านราคาและต้นทุนในการผลิตซึ่งมีอิทธิพลต่อ

คำถ้ามก้ายบท

1. Throughput และ Response Time ต่างกันอย่างไร ยกตัวอย่าง เช่น การเพิ่ม Throughput การเพิ่ม Response Time และกรณีของการเพิ่มทั้งสองอย่างพร้อมกัน
2. จากสมการ CPU Time อธิบายว่าแต่ละปัจจัยมีผลมาจากอะไรบ้าง
3. กำลังและพลังงานต่างกันอย่างไร
4. ถ้าต้องการลด Dynamic Power ลงไป 10% มีวิธีการอย่างไรบ้าง
5. จงอธิบายว่า Switching Activity มีผลต่อกำลังหรือพลังงานอย่างไร
6. จงอธิบายว่า Dynamic Power ต่างกับ Static Power อย่างไร
7. การตั้ง List Price มีหลักการอย่างไร
8. จงอธิบายขั้นตอนการผลิตชิปเบื้องต้น

สำหรับชุดคำสั่งหนึ่งที่มี CPI เคลื่อนเท่ากับ 1.2 และเครื่องที่ทำงานคำสั่งนี้มีความเร็ว 3 กิกะเอียรตซ์ เทียบกับชุดคำสั่งอีกแบบที่มี CPI เคลื่อนเท่ากับ 1.4 แต่ทำงานบนเครื่องที่มีความเร็ว 5 กิกะเอียรตซ์ พิจารณาโปรแกรมหนึ่งๆ ที่มีจำนวนคำสั่งทั้งหมด 10,000 คำสั่งในเครื่องแบบแรก และมีจำนวนคำสั่งทั้งหมด 5,000 คำสั่งในเครื่องแบบที่ 2 โปรแกรมนี้จะทำงานบนเครื่องแบบใดที่เร็วกว่ากัน และเร็วกว่ากี่เท่า

9. พิจารณาการคอมไพล์โค้ดเดียวกันออกมา 2 รูปแบบคือ 1) ประกอบด้วยแบบ A จำนวน 5 ล้านคำสั่ง และประกอบด้วยแบบ B จำนวน 4 ล้านคำสั่ง 2) ประกอบด้วยแบบ A จำนวน 2 ล้านคำสั่งและประกอบด้วยแบบ B จำนวน 7 ล้านคำสั่ง ถ้าต้องการให้โปรแกรมนี้ทำงานบนเครื่องที่มีความถี่ 1 กิกะเอียรตซ์ โค้ดแบบใดจะทำงานเร็วกว่าถ้าวัดด้วย MIPS และถ้าวัดด้วย CPU Time แบบใดจะเร็วกว่า
10. ตัววัด MIPS เหมาะสมกับการวัดประสิทธิภาพรูปแบบใดบ้าง จงยกตัวอย่างประกอบ

11. เครื่องคอมพิวเตอร์ X มีความถี่ 1.5 กิกะเฮิรตซ์ และทำงานสำหรับโปรแกรม A ได้ในเวลา 15 วินาที ในขณะเดียวกัน เมื่อนำโปรแกรม A ไปทำงานบนเครื่อง Y ได้เร็วในเวลา 10 วินาที แต่ในการทำงานบนเครื่องคอมพิวเตอร์ Y นี้ใช้จำนวน Clock มากกว่าในเครื่อง X เป็น 1.3 เท่า เครื่อง Y นี้มีความถี่เท่าไร
12. พิจารณาชุดคำสั่งที่ประกอบด้วยคำสั่งแบบ A ใช้เวลาทำงาน 3 ไซเกิล คำสั่งแบบ B ใช้เวลาทำงาน 1 ไซเกิล คำสั่งแบบ C ใช้เวลาทำงาน 2 ไซเกิล สำหรับโปรแกรม X หากใช้คอมไฟเลอร์ P ในการคอมไพล์ได้คำสั่งแบบ A จำนวน 35% คำสั่งแบบ B จำนวน 20% ที่เหลือเป็นคำสั่งแบบ C ถ้าใช้คอมไฟเลอร์ Q ในการคอมไпал์ได้คำสั่งแบบ A จำนวน 30% คำสั่งแบบ B จำนวน 50% ที่เหลือเป็นคำสั่งแบบ C จงตอบคำถามต่อไปนี้
- 12.1 รูปแบบโடดโดยคอมไฟเลอร์ใดทำงานได้เร็วกว่ากัน
 - 12.2 ถ้านำโടดทั้งสองรูปแบบไปทำงานบนเครื่องที่มีความถี่ 1.2 กิกะเฮิรตซ์ โടดแต่ละแบบ จะใช้เวลาในการทำงานเท่าไร แบบใดเร็วกว่ากัน และเร็วกว่ากันกี่เท่า
 - 12.3 ถ้าจำนวนคำสั่งทั้งหมดในโടดรูปแบบแรกเป็น 300 ล้านคำสั่ง และแบบที่ 2 เป็น 200 คำสั่ง โടดแบบใดทำงานเร็วกว่ากันในเทอมของ MIPS
13. พิจารณาโടดหนึ่งที่ประกอบด้วยประเภทต่างๆ และสำหรับสถาปัตยกรรมคำสั่งที่มี CPI ดังนี้

ประเภทคำสั่ง	ความถี่	CPI (ไซเกิล)
ประเภท A	40%	3
ประเภท B	25%	1
ประเภท C	20%	2
ประเภท D	15%	4

- 13.1 จงหาว่าเมื่อโปรแกรมนี้ทำงานจะใช้เวลาเท่าไร ถ้าทำงานบนเครื่องที่มีความถี่ 1.8 กิกะเฮิรตซ์

- 13.2 ถ้าต้องการปรับปรุงเทคโนโลยีทางฮาร์ดแวร์ให้ความถี่เพิ่มขึ้นเป็น 2.0 กิกะเฮิรตซ์ อย่างทราบว่าจะเร็วขึ้นกี่เท่า
- 13.3 ถ้าทำการปรับปรุงทางโครงสร้างให้ลด CPI ของคำสั่งประเภท A ให้เหลือ 2 แต่การปรับปรุงดังกล่าวทำให้ Cycle Time เพิ่มขึ้น 1.5 เท่า อย่างทราบว่าโปรแกรมจะทำงานได้เร็วกว่าเดิมหรือไม่ เร็วกว่าหรือช้ากว่ากี่เท่า
- 13.4 จงแนะนำวิธีที่จะทำให้โค้ดดังกล่าวทำงานเร็วขึ้น 20% ให้เสนอมา 2 วิธี อธิบายแต่ละวิธี และแสดงการคำนวณ
14. หน่วยงาน spec.org มีหน้าที่อะไร ปัจจุบันมีชุด Benchmark อะไรบ้าง จงยกตัวอย่างมาหนึ่งตัวอย่าง พร้อมอธิบายโปรแกรมที่อยู่ในชุดนั้นๆ
15. พิจารณาเครื่องคอมพิวเตอร์ดังนี้
- หน่วยความจำ DDR ขนาด 256 เมกะไบต์ ซึ่งมีความเร็ว 1 กิกะเฮิรตซ์ ขนาดหน่วยความจำเสี้ยวนานาดเท่ากับ 2 กิกะไบต์ มีฮาร์ดติสก์ขนาด 100 กิกะไบต์
- ถ้ามีการปรับปรุงเครื่องคอมพิวเตอร์ให้ทำงานที่ความถี่ 2 กิกะเฮิรตซ์ การปรับปรุงนี้มีผลให้ซึ่งมีความเร็วขึ้นเป็น 2 เท่า และการเปลี่ยนหน่วยความจำในรูปแบบ DDR เป็น DDR II และเพิ่มขนาดเป็น 512 เมกะไบต์ ซึ่งทำให้การเข้าถึงหน่วยความจำได้เร็วขึ้นโดยเฉลี่ย 40% พิจารณา Benchmark ที่มีลักษณะดังนี้
- คำสั่งเกี่ยวกับการเข้าถึงหน่วยความจำ มี 40% คำสั่งเกี่ยวกับตัวดำเนินการ ALU เท่ากับ 30% และคำสั่งอื่นๆ ที่เหลือเท่ากับ 30%
- 15.1 การอัปเกรดเครื่องในรูปแบบดังกล่าว จะทำให้ได้ Speedup โดยรวมเท่ากับเท่าไร จงอธิบาย
- 15.2 หากต้องการให้ Speedup โดยรวมเพิ่มขึ้นเป็น 2 เท่าหรือ 2.5 เท่า ทำได้หรือไม่ จงอธิบาย
16. พิจารณาการเปลี่ยนแปลงของการอปติไมโครโค้ดในลักษณะใหม่ที่จะลดการเปลี่ยนแปลงของบิตในส่วนของอปติโค้ด และจะจัดลำดับคำสั่งใหม่โดยพิจารณาการลดการเปลี่ยนแปลงของบิต

ระหว่างคำสั่ง หากการออบตีไมซ์ดังกล่าวลดการเปลี่ยนบิตระหว่าง 2 คำสั่งใดๆ ได้โดยเฉลี่ย 20% และ Power Dynamic ของการใช้โค้ดนี้จะเปลี่ยนแปลงอย่างไร และหากเปลี่ยนการใช้โวลเตจจาก 5.0 เป็น 3.5 ค่า Power Dynamic จะเปลี่ยนแปลงอย่างไร และการเปลี่ยนแปลงโวลเตจดังกล่าวมีผลกับ Energy Dynamic อย่างไร

17. พิจารณาตัวอย่างของชุด Benchmark และ SPECFP Ratio ดังตารางข้างล่าง

โปรแกรม	SPECFP Ratio
A	2015
B	3150
C	5250
D	2550
E	3210
F	1500
G	2215
H	2500

17.1 จงหาค่า Geometric Means และ Geometric Standard Deviation

17.2 ชุด Benchmark ดังกล่าวจะ Compatible กับ Lognormal Distribution หรือไม่

