บทที่ 8
หน่วยความจำแคช

หน่วยความจำแคช (cache memory) เป็นหน่วยความจำความเร็วสูงซึ่งนำมาต่อคั่นระหว่างหน่วยความจำหลักกับโพรเซสเซอร์ ดังแสดงในสไลด์หน้าที่ 4 สาเหตุเนื่องมาจากการที่หน่วยความจำหลักทำงานได้ช้ามากเมื่อเทียบกับซีพียู และเนื่องจากซีพียูต้องอ่านโปรแกรมขึ้นมาจากหน่วยความจำเพื่อที่จะปฏิบัติคำสั่งและยังต้องอ่าน/เขียนข้อมูลของคำสั่งในการปฏิบัติงานส่งผลให้ประสิทธิภาพการทำงานลดลง เนื่องจากหน่วยความจำหลักเป็นคอขวดในการส่งข้อมูลให้กับซีพียู ปัจจุบันซีพียูได้รับการพัฒนาให้เร็วได้ถึง 3.6 กิกะเฮิร์ทซ์ แต่หน่วยความจำที่เร็วที่สุดส่งข้อมูลได้ที่ความเร็วเพียง 400 MHz (หน่วยความจำ DDR400) ดังนั้นเพื่อลดเวลาที่ซีพียูจะต้องรอคอยในการอ่านข้อมูลจากหน่วยความจำแคชลง จึงมีหน่วยความจำแคชซึ่งมีความเร็วเท่ากับหรือใกล้เคียงซีพียู หน่วยความจำแคชอาจมีมากกว่า 1 ระดับได้ยกตัวอย่างในรูปของสไลด์หน้าที่ 5 จะมีแคช 2 ระดับคือระดับ 1 อยู่ภายในซีพียู และระดับที่ 2 อยู่ภายนอกซีพียู

ความจริงในทางทฤษฎีแล้วหน่วยความจำแคชไม่จำเป็นต้องอยู่ในซีพียูก็ได้ เพียงแต่ในทางปฏิบัติ ผู้ผลิตมักจะใส่หน่วยความจำแคชลงไปในซีพียูของตนเองเพื่อให้ซีพียูของตัวเองมีประสิทธิภาพที่สูงขึ้นเท่านั้น

หลักการทำงานของระบบหน่วยความจำแคชคือในการที่ซีพียูจะเข้าถึงหน่วยความจำตำแหน่งใดๆ ก็ตาม จะต้องก๊อปปี้ค่าในหน่วยความจำหลักมาใส่ในแคชก่อนแล้วค่อยให้ซีพียูอ้างถึงหน่วยความจำแคชด้วยความเร็วสูง หน่วยความจำแคชช่วยเพิ่มความเร็วการทำงานได้ เนื่องจากพฤติกรรมของโปรแกรมส่วนใหญ่เมื่อใช้หน่วยความจำตำแหน่งหนึ่งแล้ว มักจะอ่านค่าหน่วยความจำตำแหน่งนั้นๆ อีกหลายๆ ครั้ง (locality of reference) ยกตัวอย่างโค๊ดในสไลด์หน้าที่ 7 ซึ่งในโปรแกรมจะเห็นว่ามีลูปที่อ่านข้อมูลซึ่งทำการวนลูปจำนวน 100,000 ครั้ง ซึ่งกรณีนี้หน่วยความจำแคชมีประโยชน์ช่วยให้ความเร็วของระบบเพิ่มขึ้นได้มาก เพราะแม้ว่าหน่วยความจำหลักจะทำงานช้า แต่ซีพียูจะเสียเวลาในการอ่านข้อมูลขึ้นมาจากหน่วยความจำหลักใส่ในแคชแค่ครั้งเดียว และอีก 99,999 ครั้งซีพียูก็จะอ่านข้อมูลจากแคชด้วยความเร็วสูงเท่ากับความเร็วของซีพียู

ในการอ่านข้อมูลจากหน่วยความจำของโพรเซสเซอร์นั้น โพรเซสเซอร์จะไปอ่านข้อมูลในแคชก่อน หากพบข้อมูลในแคช จะเรียกว่า Cache Hit โพรเซสเซอร์ก็จะสามารถอ่านแคชได้ที่ความเร็วสูง แต่ถ้าไม่พบข้อมูลที่ต้องการในแคช จะเรียกว่า Cache Miss โพรเซสเซอร์ก็จะไปอ่านข้อมูลจากหน่วยความจำหลักมาใส่ในแคชก่อน จากนั้นจึงอ่านข้อมูลในแคชไปใช้เป็นลำดับต่อไป ในการออกแบบโพรเซสเซอร์นั้นนักออกแบบจะต้องออกแบบให้มีอัตรา Hit rate สูงสุดเท่าที่จะเป็นไปได้ ซึ่งในสไลด์หน้าที่ 8 ได้แสดงการคำนวณค่า Hit rate

การอัพเดทข้อมูลในแคช ก็มีประเด็นที่จะต้องพิจารณาเช่นเดียวกัน โดยการเขียนข้อมูลในแคชมีอยู่ 2 วิธีคือ Write back และ Write through โดยในแบบ Writeback นั้นทุกครั้งที่มีการอัพเดทค่าในหน่วยความจำจะเป็นการอัพเดทโดยการเขียนข้อมูลลงในแคชเท่านั้น แล้วค่อยไปเขียนลงหน่วยความจำหลักหลังยกเลิกการใช้แคชบล๊อกนั้นๆ ซึ่งวิธีนี้มีข้อดีคือ ประสิทธิภาพสูง แต่อย่างไรก็ตามระบบแบบนี้ ไม่เหมาะกับระบบมัลติโพรเซสเซอร์เนื่องจากบางครั้งโพรเซสเซอร์ตัวอื่นๆ ในระบบได้ใช้หน่วยความจำตำแหน่งเดียวกัน ดังนั้นหากทำการอัพเดทเฉพาะข้อมูลในแคชจะทำให้ซีพียูตัวอื่นไม่รับทราบว่าหน่วยความจำตำแหน่งนั้นๆ ถูกอัพเดทค่าไปแล้ว รูปในสไลด์หน้าที่ 10 แสดงระบบมัลติโพรเซสเซอร์ซึ่งใช้หน่วยความจำร่วมกัน วิธีการ Write through จึงเหมาะกับระบบมัลติโพรเซสเซอร์มากกว่า โดยที่ทุกครั้งที่เขียนข้อมูลในแคช ให้เขียนข้อมูลลงในหน่วยความจำหลักด้วยอย่างไรก็ตาม วิธีการ Write through จะให้ประสิทธิภาพต่ำกว่า แต่เราก็สามารถได้ความสามารถของโพรเซสเซอร์ตัวอื่นเพิ่มขึ้นมาในระบบ

ในการออกแบบหน่วยความจำแคชมี 2 วิธีเมื่อมองในแง่ของหน้าที่การใช้งานคือ Separated cache และ Unified cache โดยหน่วยความจำแคชแบบ Separated cache จะแยกส่วนของการเก็บโค๊ดคำสั่งและข้อมูลออกจากกันอย่างชัดเจนดังแสดงในสไลด์หน้าที่ 13 การมีหน่วยความจำแคชแบบแยกข้อมูลออกจากคำสั่งจะเสริมประสิทธิภาพการทำงานของซีพียูแบบไปป์ลายน์ได้เป็นอย่างดี โดยเฉพาะอย่างยิ่งในสถาปัตยกรรมซีพียูแบบ Von-Neumann ทั้งนี้เนื่องจากวงจรเฟตช์คำสั่งสามารถที่จะอ่านข้อมูลจากแคชคำสั่งได้พร้อมกับวงจร Writeback สามารถเขียนข้อมูลลงสู่แคชข้อมูลดังแสดงให้เห็นในสไลด์หน้าที่ 14

ซีพียูส่วนใหญ่จะมีหน่วยความจำแคช อยู่ 2 ระดับ คือระดับ 1 อยู่ภายในโพรเซสเซอร์ และระดับ 2 อาจอยู่ข้างในโพรเซสเซอร์หรืออยู่ภายนอกโพรเซสเซอร์ก็ได้ เช่น ในซีพียู Pentium Pro ได้แยกหน่วยความจำแคชระดับ 2 ซึ่งมีขนาด 256 กิโลไบต์-1 เมกกะไบต์ออกมาอยู่คนละชิปกับซีพียู แต่ติดตั้งอยู่บนแพกเกจเดียวกันดังแสดงในสไลด์หน้าที่ 15 ส่วนในซีพียูรุ่นใหม่ๆ ส่วนใหญ่จะรวมหน่วยความจำแคชระดับ 2 เข้ามาในชิปเดียวกับโพรเซสเซอร์แล้ว

เนื่องจากหน่วยความจำแคชมี แคชมีขนาดเล็กมาก เมื่อเทียบกับหน่วยความจำหลักดังนั้นในการใช้งานปกติหากหน่วยความจำแคชเต็ม ก็จะต้องเอาข้อมูลและคำสั่งของเก่าออกจากแคชเพื่อใส่ข้อมูลที่ต้องการ ซึ่งมีอัลกอริทึมในการตัดสินใจอยู่หลายวิธีได้แก่

- Least Recently used (LRU)
- Least Frequently Used (LFU)
- First-in-First-Out (FIFO)

อัลกอริทึมแบบ FIFO มีข้อดีคือง่ายที่สุดโดยจะนำข้อมูลที่เขียนไปครั้งแรกสุดนำมาใช้ใหม่ ซึ่งต่างจากอัลกอริทึมแบบ LRU ซึ่งจะมีการทำสถิติไว้ว่าส่วนใดไม่ได้ใช้มาเป็นเวลานานที่สุดก็จะถูก

นำออกไปจากแคช ส่วนอัลกอริทึมแบบ LFU จะมีการบันทึกค่าสถิติไว้ว่าตำแหน่งไหนใช้บ่อยน้อยที่สุดก็จะถูกนำออกไปจากแคช

การออกแบบหน่วยความจำแคช เนื่องจากหน่วยความจำแคชมีขนาดเล็กมากเมื่อเทียบกับหน่วยความจำหลัก ดังนั้นจะต้องมีวิธีการแมปหน่วยความจำหลักมาใช้ในหน่วยความจำแคช โดยมีวิธีการแมปอยู่ 3 วิธี ดังแสดงในสไลด์หน้าที่ 17

1. Direct mapping
2. Fully-associative mapping
3. Set-associative mapping

วิธี Direct mapping เป็นวิธีที่ง่ายที่สุด โดยแบ่งหน่วยความจำหลักออกเป็นกลุ่มๆ แต่ละกลุ่มมีขนาดเท่ากับหน่วยความจำแคชดังแสดงให้เห็นในสไลด์หน้าที่ 19 แสดงวิธีการแมปหน่วยความจำแคชขนาด 2 กิโลไบต์ กับหน่วยความจำหลักขนาด 64 กิโลไบต์ โดยทำการแบ่งแคชออกเป็นบล๊อกขนาดบล๊อกละ 16 ไบต์ได้จำนวน 128 บล็อก และแบ่งหน่วยความจำหลักออกเป็นบล๊อกละ 16 ไบต์ได้จำนวน 4096 บล็อก และหน่วยความจำหลักแบ่งออกเป็นกลุ่มๆ ละ 2 กิโลไบต์ กำหนดให้บล๊อกที่ 0 ของหน่วยความจำหลักของแต่ละกลุ่มสามารถใส่ค่าในแคชได้เพียงตำแหน่งเดียวคือบล๊อกที่ 0 ของแคชเท่านั้น หน่วยความจำบล๊อกอื่นๆ ของแคชก็เช่นเดียวกัน เนื่องจากแบ่งหน่วยความจำหลักได้เป็น 32 กลุ่มจึงทำการแบ่งค่าแอดเดรสออกเป็นฟิลด์ tag จำนวน 5 บิต และเนื่องจากมีจำนวนบล๊อกของหน่วยความจำแคชเท่ากับ 128 บล็อก ดังนั้นจึงแบ่งแอดเดรสออกเป็น ฟิลด์ block ได้จำนวน 7 บิต และอีก 4 บิตที่เหลือของแอดเดรสใช้อ้างอิงว่าเป็นข้อมูลไบต์ใดในบล๊อกของแคช

จากสไลด์หน้าที่ 20-21 จะเห็นว่า หน่วยความจำแคชบล๊อกที่127 เก็บข้อมูลของหน่วยความจำหลัก group2 ดังนั้นหากต้องการนำหน่วยความจำหลัก group31 บล๊อกที่ 127 มาใส่ในแคชจะต้องนำข้อมูลเก่าเขียนลงใน group2 เสียก่อนจึงอ่านค่าจาก group31 มาเขียนลงไปได้

หน่วยความจำแคชแบบ direct mapped มีข้อดีคือสามารถสร้างขึ้นมาได้ง่ายและสิ้นเปลืองวงจรสนับสนุนน้อยกว่าแคชแบบอื่นๆ อย่างไรก็ตาม หน่วยความจำแคชแบบ Direct mapped ก็มีข้อเสียคือในแคชจำนวน 128 บล็อก จะไม่สามารถมีข้อมูลในหน่วยความจำบล๊อกที่ 0, 128, 256… หรือ บล๊อกที่ 1, 129, 257 อยู่ภายในแคชในเวลาเดียวกันได้ หน่วยความจำแคชแบบ Fully-associative ออกแบบมาเพื่อแก้ไขปัญหาดังกล่าว โดยสามารถใส่บล๊อกหน่วยความจำใดๆ เข้าไปพร้อมกับบล๊อกใดๆ ก็ได้ ดังแสดงให้เห็นในสไลด์หน้าที่ 23 จะเห็นว่าเนื่องจากหน่วยความจำหลักมีจำนวน 4095 บล็อก ดังนั้นจำนวน tag bit ในแคชแต่ละบรรทัดจึงมีเท่ากับ 12 บิต จากตัวอย่างมีแคชจำนวน 128 บล็อก ดังนั้นจึงสิ้นเปลืองบิตของ tag ไปเท่ากับ 12*128 = 1,536 บิต ซึ่งถือว่าสูงมาเมื่อเทียบกับจำนวนบิตของ Direct mapped ซึ่งใช้จำนวนบิตของ tag ไปเท่ากับ 5*128 = 640 บิต นอกจากนี้หน่วยความจำแคชแบบ Fully associative ยังต้องการวงจรเปรียบเทียบเพื่อหาว่ามีข้อมูลใน

แคชหรือไม่ขนาดใหญ่มาก ดังนั้นในโพรเซสเซอร์ส่วนใหญ่ในท้องตลาดจึงไม่ค่อยใช้วิธีแมปวิธีนี้มากนัก แคชแบบ Fully-associative ที่แสดงให้เห็นในสไลด์นี้แมปเข้ากับโพรเซสเซอร์ที่สามารถอ้างหน่วยความจำได้สูงสุด 64 กิโลไบต์และมีแคชขนาด 2 กิโลไบต์ (ไม่รวม tag bit)

แคชแบบ Set-associative จะรวมเอาข้อดีของแคชแบบ Direct mapped และแบบ Fully-associative เข้าด้วยกัน สไลด์หน้าที่ 26 แสดงหน่วยความจำแคชของโพรเซสเซอร์ซึ่งสามารถอ้างหน่วยความจำหลักได้ 64 กิโลไบต์และมีหน่วยความจำแคชขนาด 2 กิโลไบต์ ทำการแบ่งแคชออกเป็นบล๊อกโดยมีขนาดบล๊อกละ 16 ไบต์ได้แคชจำนวน 2 ทาง มีทางละ 64 เซ็ท จึงเรียกว่าแคชแบบ 2-way set associative mapped จากนั้นทำการแบ่งหน่วยความจำหลักออกเป็นกลุ่ม กลุ่มละ 64 บล๊อกได้ทั้งหมด 64 กลุ่ม ดังนั้นจำนวนบิตที่ต้องใช้ใน tag field เท่ากับ 6 บิตและจำนวนบิตที่ต้องใช้ใน block field เท่ากับ 6 บิตและจำนวนบิตที่ใช้ระบุว่าเป็นข้อมูลไบต์ใดในบล๊อกมีขนาดเท่ากับ 4 บิต

สไลด์หน้าที่ 28 แสดงหน่วยความจำแคชแบบ 4-way set associative โดยแบ่งแคชออกเป็น 4 ทาง ทางละ 32 เซ็ท และแบ่งหน่วยความจำหลักออกเป็นกลุ่ม กลุ่มละ 32 บล็อก ได้ทั้งหมดได้ 128 กลุ่ม ดังนั้นจำนวนบิตของ tag จะเท่ากับ 7 บิต

เนื่องจากหน่วยความจำแคชจะต้องมีความเร็วสูง ดังนั้นหน่วยความจำแคชของซีพียูส่วนใหญ่จึงสร้างมาจากหน่วยความจำแบบสเตติก ซึ่งต้องการจำนวนทรานซิสเตอร์ในการสร้างจำนวนมาก จากที่เราเรียนมาในบทที่ 4 เรื่องหน่วยความจำพบว่าหน่วยความจำแบบสเตติกแรมต้องใช้ทรานซิสเตอร์จำนวน 6 ตัวในการสร้างเซลล์ของหน่วยความจำจำนวน 1 บิต จากตารางที่ 8.1 เราจะเห็นได้ว่าหน่วยความจำแคชของซีพียู Athlon64 มีขนาดเท่ากับ 128K+1M ดังนั้นจำนวนทรานซิสเตอร์คร่าวๆ ที่ใช้ในการสร้างแคชจะมีมากถึง (128*1024*8*6)+ (1*1024*1024*8*6)= 56,623,104 ตัวซึ่งจำนวนนี้ไม่นับวงจรสนับสนุนหน่วยความจำแคชซึ่งต้องมีเพิ่มขึ้นมาอีกเช่น tag bit เป็นต้น จากตารางในสไลด์หน้าที่ 30 แสดงให้เราเห็นว่าซีพียู Athlon64 มีจำนวนทรานซิสเตอร์เท่ากับ 105.9 ล้านตัว เป็นจำนวนทรานซิสเตอร์ของหน่วยความจำแคชถึง 56 ล้านตัวคิดเป็น 53.46 เปอร์เซนต์ของจำนวนทรานซิสเตอร์ที่ซีพียูใช้
