บทที่ 2
รูปแบบของข้อมูลในคอมพิวเตอร์

ในการจัดเก็บข้อมูลตัวเลขของคอมพิวเตอร์ เราสามารถเก็บได้สองรูปแบบคือตัวเลขจำนวนเต็มและตัวเลขทศนิยม โดยการเก็บตัวเลขแบบจำนวนเต็มนั้นเราสามารถเก็บได้ทั้งแบบมีเครื่องหมายและแบบไม่มีเครื่องหมาย ยกตัวอย่างเช่นตัวเลขขนาด 8 บิต หากเก็บเลขแบบไม่มีเครื่องหมายจะสามรถเก็บได้ตั้งแต่ค่า 0-255 แต่ถ้าเก็บแบบมีเครื่องหมายก็จะแล้วแต่วิธีการเก็บว่าเราจะใช้วิธีการเก็บข้อมูลแบบไหน สไลด์หน้าที่ 5 แสดงให้เห็นวิธีการเก็บตัวเลขฐานสองแบบมีเครื่องหมายสามรูปแบบด้วยกันอันได้แก่แบบ sign and magnitude และแบบ 1's complement และแบบ 2's complement การเก็บแบบ 2's complement จะนิยมใช้มากที่สุดเนื่องจากง่ายต่อการออกแบบวงจรสนับสนุนการทำโอเปอเรชั่นทางคณิตศาสตร์ดังแสดงในรูปสไลด์หน้าที่ 8 ดังนั้นหากนักศึกษาไปอ่านหนังสือเล่มใดที่บอกว่าเก็บตัวเลขเป็นตัวเลขฐานสองแบบมีเครื่องหมายแล้วไม่มีการบอกว่าเก็บด้วยวิธีใดแล้ว ให้ถือว่าเป็นแบบ 2's complement เอาไว้ก่อน และเนื่องจากการเก็บตัวเลขฐานสองแบบ 2's complement นี้มีข้อจำกัดของค่าที่จะเก็บได้อยู่ที่จำนวนบิตของตัวเลขที่เก็บ ดังนั้นหากนำตัวเลขแบบมีเครื่องหมายแบบ 2's complement มาทำโอเปอเรชั่นทางคณิตศาสตร์กันจะมีโอกาสที่ข้อมูลผลลัพธ์ที่ได้จะมีค่ามากหรือน้อยเกินไปที่ตัวเลขขนาดจำนวนบิตนั้นๆ สามารถเก็บได้ เราเรียกสถานะดังกล่าวว่า Overflow

สไลด์หน้าที่ 10 แสดงถึงการตรวจเช็คสถานะ Overflow ของการทำการบวกเลข โดยสถานะ Overflow จะเกิดขึ้นก็ต่อเมื่อทำการบวกเลขที่มีเครื่องหมายเหมือนกันและได้ผลลัพธ์มีเครื่องหมายต่างไปจากตัวตั้งเท่านั้น ดังนั้นสมการในการตรวจเช็คโอเวอร์โฟลว์ของการบวกเลข X และ Y และได้ผลลัพธ์เท่ากับ R คือ $Ov = (\overline{XS}\ \overline{YS}\ RS) + (XS\ YS\ \overline{RS})$ โดยกำหนดให้ Xs, Ys, Rs เป็นบิต MSB ของ X, Y และ R ตามลำดับ

ในการทำโอเปอเรชั่นของตัวเลขแบบ 2's complement ที่มีขนาดแตกต่างกันนั้นเราจะต้องแปลงตัวเลขที่มีจำนวนบิตต่ำกว่าให้มีจำนวนบิตเท่ากันเสียก่อนที่จะทำโอเปอเรชั่นที่ต้องการ ได้ เราเรียกการขยายจำนวนบิตนี้ว่าการทำ Sign extension ซึ่งอัลกอริทึมในการทำ Sign extension นี้แสดงให้เห็นในสไลด์หน้าที่ 12 โดยจะทำการเติมบิตเครื่องหมายเข้าไปข้างหน้าของเลขที่ต้องการขยายจำนวนบิตให้ได้จำนวนบิตที่ต้องการ ซึ่งการทำ sign extension นี้จะต้องใช้ในหลายวงจรด้วยกันซึ่งนักศึกษาจะได้เรียนถึงตัวอย่างการใช้งานต่อไปในบทที่ 3

สไลด์หน้าที่ 14 แสดงให้เห็นถึงหน่วยของการอ้างอิงข้อมูลในคอมพิวเตอร์ โดยเราจะถือว่าข้อมูลขนาด 8 บิตจะเท่ากับ 1 ไบต์ ส่วนข้อมูลขนาด 1 เวิร์ด (Word) จะมีขนาดตั้งแต่ 16 บิตขึ้นไป ซึ่งขนาดของเวิร์ดนี้ไม่จำเป็นต้องมีค่าเท่ากับ 16 บิตเพียงอย่างเดียว แต่ขนาดของเวิร์ดมีตั้งแต่ 16-64 บิต ขึ้นอยู่กับว่าสถาปัตยกรรมของซีพียูที่ใช้จะระบุว่าข้อมูล 1 เวิร์ดกว้างเท่าใด โดยปกติซีพียูส่วนใหญ่จะ

ทำการอ่านข้อมูลจากหน่วยความจำครั้งละ 1 เวิร์ดไม่ว่าจะอ่านโปรแกรมหรืออ่านข้อมูลก็ตาม ดังนั้นในซีพียูขนาด 32 บิตแล้ว การอ่านข้อมูล 1 ครั้งจากหน่วยความจำจะได้ข้อมูลขนาด 4 ไบต์หรือ 1 เวิร์ดนั่นเอง สไลด์หน้าที่ 16 แสดงให้เห็นถึงการเก็บตัวอักษรด้วยรหัสแอสกี้จำนวน 4 ตัวสามารถเก็บค่าเอาไว้ในข้อมูลขนาด 1 เวิร์ดได้ หรือในกรณีที่เรามองค่าขนาด 32 บิตเป็นตัวเลขแบบ 2's complement ก็จะได้ตัวเลขขนาด $(2^{31}-1)….2^{31}$

จากที่ได้กล่าวมาแล้วว่า Stored program concept นั้นเราจะต้องเก็บข้อมูลเอาไว้ในหน่วยความจำ ระบบคอมพิวเตอร์ส่วนใหญ่มักจะกำหนดให้หน่วยความจำ 1 ตำแหน่งเก็บข้อมูลได้จำนวน 8 บิตหรือ 1 ไบต์ หรือที่เราเรียกว่า Byte addressable addressing แต่เนื่องจากซีพียูส่วนใหญ่อ่านข้อมูลครั้งละ 1 เวิร์ด ซึ่งค่า 1 เวิร์ดจะมีค่าเท่ากับขนาดของซีพียู ดังนั้นการอ่านข้อมูล 1 ครั้งของซีพียูจะสามารถได้ค่าจากหน่วยความจำหลายตำแหน่งก็ได้ เช่นซีพียู 32 บิต อ่านข้อมูลจากหน่วยความจำ 1 ครั้งจะได้ข้อมูลจากหน่วยความจำจำนวน 4 ตำแหน่ง เป็นต้น

เนื่องจากหน่วยความจำเราจัดโครงสร้างเป็นแบบ byte addressable แต่ซีพียูอ่านข้อมูลขนาด 32 บิต ดังนั้นจะต้องมีการระบุว่าข้อมูลใน 4 ไบต์นั้นไบต์ใดมีลำดับความสำคัญสูงสุดไบต์ใดมีความสำคัญรองลงมา และไบต์ใดมีความสำคัญต่ำสุด ในสไลด์หน้าที่ 19 เป็นการอธิบายถึงการจัดเรียงค่าของไบต์ของหน่วยความจำในการเก็บค่าตัวเลขจำนวนมากกว่า 1 ไบต์ในหน่วยความจำซึ่งมีอยู่ 2 แบบได้แก่แบบ Big Endian และแบบ Little Endian โดยซีพียูที่ใช้แบบ Big Endian จะได้แก่ 68k หรือ 68000 ของโมโตโรล่า, SPARC ของ Sun, MIPS ของ Silicon Graphics และ PA-RISC ของ HP ส่วนสถาปัตยกรรมที่ใช้แบบ Little Endian จะได้แก่ 80x86 ของอินเทล และ Alpha ของ Compaq เป็นต้น

ในการเก็บค่า word ใส่ในสถาปัตยกรรมแบบ Big-endian นั้นจะใช้แอดเดรสไบต์ต่ำสุดเก็บค่าของข้อมูลไบต์สูงสุดของเวิร์ด ในขณะที่สถาปัตยกรรมแบบ Little-endian จะเก็บไบต์ต่ำของเวิร์ดด้วยแอดเดรสไบต์ต่ำและเก็บค่าไบต์สูงของเวิร์ดไว้ในแอดเดรสไบต์สูง

รูปในสไลด์หน้าที่ 20 นั้นจะแสดงถึงการเก็บข้อมูลในแบบ big-endian และแบบ little-endian ว่าทั้งสองแบบมีความแตกต่างกันอย่างไร โดยกำหนดให้มีการเก็บค่า 201F539AH ซึ่งเป็นตัวเลขขนาด 32 บิต ซึ่งจะเห็นว่ามีรูปแบบการเรียงไบต์ของข้อมูลที่ต่างกันอยู่ หากมองตามรูปในสไลด์หน้าที่ 20 จะเห็นว่าแบบ little-endian มองง่ายกว่า แต่อย่างไรก็ตามหากเรากลับลำดับการเรียงตำแหน่งแอดเดรสเสียใหม่เป็นดังรูปในสไลด์หน้าที่ 21 แบบ big-endian จะมองง่ายกว่า

สไลด์หน้าที่ 22 แสดงให้เห็นถึงเรื่องของการจัดเรียงข้อมูลที่ต้องการให้อยู่ภายใน word address ซึ่งจะเห็นว่ารูปด้านซ้ายมือนั้นเราเก็บข้อมูลขนาด 32 บิตไว้ใน word address แค่แอดเดรสเดียว ซึ่งเราเรียกว่าข้อมูลนี้ aligned อยู่ในเวิร์ดเอดเดรส แต่ในรูปทางด้านขวามือจะเห็นว่าข้อมูล 1 เวิร์ดที่เก็บอยู่แม้จะใช้ 4 ไบต์แอดเดรสในการเก็บเท่ากันกับแบบแรก แต่ใช้เวิร์ดแอดเดรสในการเก็บ 2

แอดเดรสด้วยกัน เราเรียกว่าข้อมูลนั้นไม่ aligned อยู่ใน เวิร์ดแอดเดรส ในซีพียูส่วนใหญ่จะกำหนดไว้ชัดเจนว่าข้อมูลขนาด 1 เวิร์ดจะต้องเรียงอยู่ใน word address เพียงแอดเดรสเดียวเท่านั้น แต่ก็มีซีพียูบางตระกูลอนุญาตให้เก็บแบบ unaligned address ได้

สไลด์หน้าที่ 23 แสดงให้เห็นถึงการเก็บตัวเลขทศนิยมแบบ Fixed point number โดยบิตหน้าสุดเก็บค่า sign bit โดยหากมีค่าเป็น 1 จะแสดงว่าเป็นค่าลบและหากเป็น 0 จะเป็นค่าบวก บิตถัดมาจะเป็นค่าหลังเลขทศนิยมดังแสดงให้เห็นในสมการ สไลด์หน้าที่ 24 แสดงตัวอย่างการเก็บตัวเลขโดยใช้วิธี Fixed point อย่างไรก็ดีวิธีการเก็บเลขทศนิยมแบบ fixed point มีข้อจำกัดอยู่มากตรงที่สามารถแทนค่าได้ค่อนข้างอยู่ในย่านที่แคบมาก ยกตัวอย่างเช่นเลข fixed point ขนาด 32 บิตจะสามารถแทนค่าตัวเลขได้ตั้งแต่ 0…$\pm 2.15*10^{9}$ ซึ่งเราจะเห็นว่ามีค่าน้อยมาก ซึ่งไม่เพียงพอที่จะเก็บตัวเลขที่ต้องใช้ในงานทางด้านวิทยาศาสตร์ได้ เช่น ค่าตัวเลข Avogadro หรือค่าคงที่ของ Planck เป็นต้น ดังนั้นจึงมีการคิดค้นวิธีในการแทนตัวเลขทศนิยมในคอมพิวเตอร์ขึ้นมาใหม่ ซึ่งก็คือตัวเลขแบบ Floating point ดังแสดงในสไลด์หน้าที่ 25-26 นั่นเอง

สไลด์หน้าที่ 27 แสดงฟอร์แมตของตัวเลข Floating point มาตรฐาน IEEE 754 ซึ่งกำหนดให้ตัวเลขมีขนาดเท่ากับ 32 บิต กำหนดให้บิตหน้าสุดเป็นบิตแสดงเครื่องหมาย โดยหากมีค่าเป็น 1 จะแสดงว่าเป็นค่าลบและหากเป็น 0 จะเป็นค่าบวก อีก 8 บิตถัดมาจะเป็นค่า Exponent และอีก 23 บิต จะเก็บค่า mantissa ซึ่งค่าตัวเลขที่เก็บจะมีค่าเท่ากับ $\pm 1.M \times 2^{E'-127}$ กค่าที่จะเก็บอยู่ในฟอร์แมตนี้จะต้องเป็นค่าที่เรียกว่า normalized value เท่านั้น ซึ่งค่า normalized คือค่าที่มีบิตหน้าสุดของตัวเลขเท่ากับค่า 1 เช่นตัวเลข $0.0101011110*2^{10}$ จะเป็นเลขแบบ unnormalized ซึ่งเราจะต้องแปลงค่าดังกล่าวให้เป็นค่า $1.01011110*2^{8}$ ก่อนจึงจะเก็บลงในฟอร์แมตนี้ได้ โดยในการเก็บค่านั้นเราจะเก็บแค่บิตที่ตามหลังจุดทศนิยมใส่ในฟิลด์ mantissa โดยไม่ต้องเก็บบิตหน้าทศนิยมเพราะถือว่าค่าหน้าบิตทศนิยมต้องเป็นค่า 1 เสมอ หรือเป็นค่า normalized แล้วเสมอนั่นเอง ในสไลด์หน้าที่ 30 แสดงให้เห็นถึงตัวอย่างการเก็บค่า $1.0010110….1 \times 2^{-87}$ ลงในฟอร์แมต IEEE 754 ส่วนสไลด์หน้าที่ 31-32 แสดงให้เห็นถึงตัวอย่างการแปลงค่า unnormalized value ให้กลายเป็นค่า normalized value

ฟิลด์ exponent จะเก็บค่ายกกำลังซึ่งค่าที่เก็บจะเป็นเลขฐานสองแบบมีเครื่องหมายค่าตั้งแต่ -126…+127 โดยค่าตัวเลขที่เก็บจะเป็นตัวเลขแบบรหัสเกิน 127 สาเหตุที่เก็บด้วยรหัสแบบนี้เนื่องจากทำให้ง่ายต่อการสร้างวงจรเปรียบเทียบค่าในการทำโอเปอเรชั่นทางคณิตศาสตร์ของเลขทศนิยม ดังนั้นค่าที่เก็บอยู่ในฟิลด์นี้หากเราต้องการรู้ว่าเก็บค่ายกกำลังอะไรเราจะต้องนำค่าในฟิลด์ exponent นี้มาลบออกด้วยค่า 127 ก่อน สไลด์หน้าที่ 29 แสดงความหมายของฟิลด์ Exponent ในกรณีที่ค่าของตัวเลขที่เก็บมีค่าเท่ากับ 0 หรือ 255 โดยหากค่าฟิลด์exponent มีค่าเป็น ศูนย์แล้ว หากค่าในฟิลด์ mantissa มีค่าเท่ากับ 0 ทุกบิตจะหมายถึงค่า 0.0000000000…0 แต่ถ้าค่าในฟิลด์ exponent มีค่าเท่ากับ 255 แล้ว หาก

ค่าในฟิลด์ mantissa มีค่าเท่ากับ 0 จะหมายถึงค่า infinity แต่ถ้าค่าในฟิลด์ mantissa ไม่เท่ากับ 0 แสดงว่าค่าที่เก็บไม่ใช่ค่าตัวเลขที่ถูกต้อง (NaN : Not A Number) เช่น การหารตัวเลขด้วยศูนย์ (Divide by zero) เป็นต้น

สไลด์หน้าที่ 33 แสดงอัลกอริทึมในการบวกลบเลข Floating point โดยในการบวกลบเลขนั้นก่อนอื่นเราจะต้องทำให้ค่า exponent ของเลขทั้งสองเท่ากันก่อน หากไม่เท่ากันก็จะเลือกเอาตัวเลขที่มีค่า exponent ต่ำกว่ามาปรับให้มีค่า exponent เท่ากับอีกตัวหนึ่ง จากนั้นจึงทำการบวกลบค่าในฟิลด์ mantissa เข้าด้วยกันก็จะได้ผลลัพธ์ แต่ก่อนที่จะเก็บค่าผลลัพธ์ได้นั้นจะต้องเช็คว่าผลลัพธ์มีค่าเป็นเลข normalized value แล้วหรือไม่ หากไม่ก็จะต้องปรับให้เป็นค่า normalized value ก่อนจึงจะจัดเก็บค่าผลลัพธ์ได้ ในสไลด์หน้าที่ 34 แสดงให้เห็นตัวอย่างการบวกเลข โดยค่าที่แสดงให้เห็นจะเป็นเลขฐานสิบเพื่อให้ง่ายต่อการทำความเข้าใจ

สไลด์หน้าที่ 35 แสดงอัลกอริทึมการคูณเลข Floating point โดยจะต้องทำการนำค่าในฟิลด์ exponent มาบวกกันแล้วลบด้วย 127 แล้วเก็บผลลัพ์ของเทอม exponent ส่วนผลลัพธ์ในเทอม mantissa ก็ได้จากการนำค่าในเทอม mantissa มาคูณกัน แต่ก่อนที่จะเก็บค่าผลลัพธ์ได้นั้นจะต้องเช็คว่าผลลัพธ์มีค่าเป็นเลข normalized value แล้วหรือไม่ หากไม่ก็จะต้องปรับให้เป็นค่า normalized value ก่อนจึงจะจัดเก็บค่าผลลัพธ์ได้ ในสไลด์หน้าที่ 36 แสดงให้เห็นตัวอย่างการบวกเลข โดยค่าที่แสดงให้เห็นจะเป็นเลขฐานสิบเพื่อให้ง่ายต่อการทำความเข้าใจ

สไลด์หน้าที่ 37 แสดงอัลกอริทึมการหารเลข Floating point ซึ่งจะคล้ายกับการคูณมากเพียงแต่เปลี่ยนการบวกค่า exponent เป็นการลบค่า exponent และทำการหารค่า mantissa เข้าด้วยกัน และทำการ normalize ผลลัพธ์หากจำเป็น
