

สถาปัตยกรรมคอมพิวเตอร์ Computer Architecture

การออกแบบและการวิเคราะห์
Design and Analysis



ดร. เกरिक พิรมย์โสภา

Krerik Piromsopa, Ph.D.

ฉบับ พ.ศ. 2560

สถาปัตยกรรมคอมพิวเตอร์

Computer Architecture

การออกแบบและการวิเคราะห์
Design and Analysis

ดร. เกริก ภิรมย์โสภา
Krerik Piromsopa, Ph. D.

ฉบับปรับปรุง พ.ศ. 2559

สำหรับ คุณภกาทิพย์ เด็กชายดีดี และ เด็กหญิงลดา
ขอบคุณสำหรับความหมายของชีวิต

คำนำ

สถาปัตยกรรมคอมพิวเตอร์ (ภาษาอังกฤษใช้คำว่า computer architecture) เป็นพื้นฐานสำคัญที่นักคอมพิวเตอร์ทุกคนต้องรู้ บ่อยครั้งมักจะมีผู้โต้แย้งว่า แม้จะไม่มีความเข้าใจอันดีในสถาปัตยกรรมคอมพิวเตอร์ เราก็คงยังเป็นนักคอมพิวเตอร์ได้ คำกล่าวนี้คงจะไม่จริงนัก เพราะทุกอย่างที่เป็นฮาร์ดแวร์ของคอมพิวเตอร์ ตั้งแต่คำสั่งที่ใช้ได้ การจัดเก็บข้อมูล การประมวลผล คือสิ่งที่กำหนดโดยสถาปัตยกรรมคอมพิวเตอร์ทั้งหมด ในฐานะวิศวกรคอมพิวเตอร์ และ นักวิทยาศาสตร์คอมพิวเตอร์ นั้น การออกแบบ และการพัฒนา ระบบสถาปัตยกรรมคอมพิวเตอร์เป็นหน้าที่หลักโดยตรง ในฐานะนักพัฒนาซอฟต์แวร์ แม้จะมีได้ออกแบบหรือพัฒนาสถาปัตยกรรมคอมพิวเตอร์โดยตรง แต่ความเข้าใจอันดีในสถาปัตยกรรมคอมพิวเตอร์ ย่อมช่วยให้สามารถพัฒนาซอฟต์แวร์ได้ดีขึ้น

เพื่อให้เห็นภาพชัดเจนยิ่งขึ้น จึงขอยกตัวอย่างประกอบดังนี้ ในช่วงที่ผมกำลังศึกษาระดับปริญญาเอกในต่างประเทศนั้นมีการเรียนการสอนวิชาอัลกอริทึม (algorithm) ซึ่งอาจารย์ผู้สอนในตอนนั้น ได้ให้ผู้เรียนแข่งขันทำโปรแกรมเป็นโครงการรายวิชาเพื่อแก้ปัญหาเอ็นพีบริบูรณ์ (NP-Complete) ให้ได้เร็วที่สุด หากใครเคยศึกษาปัญหาเอ็นพีบริบูรณ์มาบ้าง ย่อมทราบว่าปัญหาในลักษณะดังกล่าวไม่สามารถหาคำตอบได้โดยใช้เวลาเป็นพหุนาม กล่าวคือการหาคำตอบที่เหมาะสมนั้นต้องทดสอบวัดค่าจากทุกกรณีที่เป็นไปได้เพื่อให้ได้ทราบคำตอบที่ดีที่สุด เมื่อทราบเช่นนั้น ผมก็คาดเดาได้ไม่ยากว่า โปรแกรมของเพื่อนร่วมชั้นเรียนแต่ละคน คงจะมีลักษณะไม่แตกต่างกันมาก เพราะแนวทางคร่าว ๆ คือ ต้องทำการวนซ้ำเพื่อวิเคราะห์กรณีที่เป็นไปได้ทั้งหมด และ นำคำตอบที่ดีที่สุดออกมาแสดง ในการแข่งขันครั้งนี้ผมได้ใช้ความรู้พื้นฐานด้านสถาปัตยกรรมคอมพิวเตอร์ของตนเอง เลือกใช้โครงสร้างข้อมูลแบบที่เหมาะสมและสามารถประมวลผลได้เร็วบนสถาปัตยกรรมแบบ Intel Pentium 3 และ Intel Pentium 4 (ซึ่งเป็นสถาปัตยกรรมที่ดีที่สุดในขณะนั้น) เอาชนะกลุ่มเพื่อนหลายคนในชั้นเรียน จนทำเวลาได้เป็นอันดับที่สองของชั้นเรียน ซึ่งเพื่อนร่วมชั้นเรียนต่างก็เป็นนักศึกษาปริญญาโทและปริญญาเอกที่มีความสามารถสูง ในครั้งนั้นเพื่อนร่วมชั้นที่ได้ผลการทำงานเป็นที่หนึ่งใช้เทคนิคที่แตกต่างกันไปกับวิธีการที่ผมใช้ ซึ่งซับซ้อนกว่า เข้าใจได้ยาก และ ในบางกรณีทำงานได้ช้ากว่ามาก แต่เมื่อนับเวลาโดยรวมแล้วสามารถทำได้เร็วกว่า จึงเป็นผู้ชนะไป

ประสบการณ์ในครั้งนั้นชี้ให้เห็นว่า หากมีความเข้าใจในสถาปัตยกรรมคอมพิวเตอร์ดีพอ แม้การเขียนโปรแกรมง่าย ๆ ในภาษาระดับสูง ก็สามารถจะดึงสมรรถนะของหน่วยประมวลผลกลางออกมาใช้ได้ทุกหยด ต่างจากเพื่อนหลายคนในห้องเรียนครั้งนั้น ที่เขียนโปรแกรมลักษณะเดียวกันหรือคล้ายคลึงกับของผม แต่เนื่องจากความเข้าใจในสถาปัตยกรรม

คอมพิวเตอร์อาจไม่ลึกซึ้งมากพอ จึงไม่สามารถที่จะดึงความสามารถบางอย่างของหน่วยประมวลผลออกมาใช้ได้ ทำให้ได้สมรรถนะที่ได้แก่กว่าของมมมาก แน่นอนว่าผมมีประสบการณ์ที่คล้ายกันนี้ในหลายงานที่ผมเคยมีส่วนเกี่ยวข้อง สิ่งที่ยกขึ้นมานี้เป็นเพียงกรณีศึกษาอันหนึ่งเท่านั้น (เคยมีนิสิตมาถามผมว่าอัลกอริทึมที่เค้าใช้ก็เหมือนกับของผม แต่ทำไมของเค้าถึงช้ากว่าของมมมาก ผมมอม่ยมใจแล้วตอบนิสิตว่า อีกครั้งหนึ่งอยู่ที่โครงสร้างข้อมูลทีเลือกใช้ และการเลือกใช้คำสั่งทีเหมาะสมด้วยครับ)

ดังนั้นผมจึงอยากใช้หนังสือเล่มนี้ส่งข้อความถึง นิสิต นักศึกษา และ นักเรียนทั้งหลาย ว่าความเข้าใจอันดีในสถาปัตยกรรมคอมพิวเตอร์ นอกจากจะเป็นพื้นฐานสำคัญสำหรับการศึกษาด้านคอมพิวเตอร์ ยังเป็นสิ่งจำเป็นสำหรับการทำงานในสาขาวิชาชีพทางด้านคอมพิวเตอร์ทุกสาขา ผมหวังว่า การได้อ่านหนังสือทีดี การได้ทดลอง(ออกแบบ) รวมถึงการวิเคราะห์ทีดี จะช่วยให้สามารถศึกษาได้อย่างต้องแท้มากขึ้น

หนังสือเล่มนี้พยายามวางพื้นฐานทีสำคัญ เริ่มจากความเข้าใจในสถาปัตยกรรมคอมพิวเตอร์ ไปถึงการวิเคราะห์สถาปัตยกรรมคอมพิวเตอร์ทีหลากหลายแบบต่าง ๆ อย่างไรก็ตาม ศาสตร์ด้านนี้มีการเปลี่ยนแปลงอยู่ตลอดเวลา และ สถาปัตยกรรมทีถูกมองว่าดีในเวลาหนึ่ง มักจะถูกหักล้างในเวลาถัดมาด้วยเทคโนโลยีและความต้องการทีเปลี่ยนไป ผมจึงพยายามจัดวางเนื้อหา ให้เหมาะกับการสอนในหนึ่งภาคการศึกษา ซึ่งสอดคล้องกับหลายวิชา เช่น Computer System Architecture ซึ่งเป็นวิชาบังคับในหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ ของ ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย (ในบางหลักสูตรอาจจะใช้ชื่อวิชาว่า Computer Organization and Architecture) ซึ่งอาจจะไม่ครอบคลุมถึงสถาปัตยกรรมทุกรูปแบบ รวมถึงเนื้อหาในบางด้าน อาจจะไม่ลงรายละเอียดในเชิงลึกมากนัก แต่จะเน้นในเชิงวิเคราะห์และการนำไปใช้เป็นหลัก ในบางช่วงบางตอนที่ต้องการให้เห็นโครงสร้าง ก็จะทำการอธิบายด้วยแผนภาพ (ในการปรับปรุงหนังสือเล่มนี้ครั้งถัดไป ผู้เขียนวางแผนจะใช้ภาษา Verilog HDL และใช้การ Simulation เพื่อประกอบความเข้าใจ)

ทั้งนี้ขอออกตัวว่า ผมมิได้มองว่าตัวเองเป็นผู้เชี่ยวชาญด้านสถาปัตยกรรมคอมพิวเตอร์แต่อย่างใด (แน่นอนว่าคนที่ทำงานในสายวิชาชีพนี้ หากบอกว่าตัวเองเชี่ยวชาญในด้านใด คงจะเป็นความคิดทีไม่ค่อยดีนัก เพราะศาสตร์ด้านนี้เปลี่ยนแปลงรวดเร็วทุกวันจนสิ่งที่เราเรียนรู้พร้อมจะตกุ่นได้ทีละเวลา) แม้ผมจะมีงานวิจัยและได้เคยยื่นขอจดสิทธิบัตร (pending) เกี่ยวกับสถาปัตยกรรมคอมพิวเตอร์ทีมีความสามารถด้านความมั่นคงปลอดภัยอยู่บ้างก็ตาม หากแต่ผมเพียงอยากเห็นนักคอมพิวเตอร์รุ่นใหม่ (โดยเฉพาะกลุ่มทีเป็นคนไทย) เข้าใจสถาปัตยกรรมคอมพิวเตอร์และนำไปประยุกต์ใช้ประโยชน์ เพื่อให้สามารถพัฒนาโปรแกรมทีมีสมรรถนะมากขึ้น หรืออย่างน้อย ก็สมารถทีจะเลือกใช้สถาปัตยกรรมคอมพิวเตอร์ทีเหมาะสมกับงานทีต้องการใช้ได้

จากความชอบส่วนตัวของผม และ ประสบการณ์ที่เกี่ยวข้องกับการเรียนการสอนด้าน ฮาร์ดแวร์และสถาปัตยกรรมคอมพิวเตอร์กว่าสี่ปี (ผมเริ่มสอนวิชาสถาปัตยกรรมคอมพิวเตอร์ครั้งแรกในฐานะอาจารย์พิเศษที่มหาวิทยาลัยเอกชน ตั้งแต่ยังเรียนปริญญาโท) ผมจึงอยากถ่ายทอดประสบการณ์ และ ความรู้ ที่ส่วนหนึ่งได้จากค้นคว้าและการลองถูกลองผิด ให้กับ นิสิต นักศึกษาและคนรุ่นต่อไปได้ทราบ โดยหวังว่านอกจากจะเป็นการช่วยให้ นิสิต นักศึกษา สามารถเรียนลัดได้เร็วขึ้นแล้ว ยังจะเป็นการสืบต่อองค์ความรู้ด้าน สถาปัตยกรรมคอมพิวเตอร์ (ซึ่งปัจจุบัน นักพัฒนาซอฟต์แวร์รุ่นใหม่มักไม่ค่อยให้ความสนใจหรือไม่เห็นความสำคัญ) ให้เป็นพื้นฐานสำคัญ และยังคงอยู่ต่อไป

สุดท้ายนี้ผู้เขียนขอขอบคุณท่าน รองศาสตราจารย์ ดร. สมชาย ประสิทธิ์จตุระกุล ที่ได้ให้ คำแนะนำด้านการเขียนและการจัดพิมพ์ หนังสือเล่มนี้ประสบความสำเร็จได้ส่วนหนึ่งเพราะ คำแนะนำของท่านครับ

ดร. เกริก ภิรมย์โสภา

ผู้ช่วยศาสตราจารย์
ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย

Krek.P@chula.ac.th

1 ธันวาคม 2559

สารบัญ

1	บทนำ.....	1
1.1	รู้ก่อนเรียน.....	1
1.2	ส่วนประกอบของระบบคอมพิวเตอร์.....	2
1.3	คำจำกัดความ.....	3
1.4	สถาปัตยกรรมชุดคำสั่ง (Instruction Set Architecture).....	4
1.5	โครงสร้างคอมพิวเตอร์ (Computer Organization).....	6
1.6	การสร้างระบบคอมพิวเตอร์ (Implementation).....	7
1.7	เป้าหมายของการออกแบบ.....	8
1.8	แบบฝึกหัดท้ายบท.....	9
2	สมรรถนะของระบบคอมพิวเตอร์.....	11
2.1	การวัดสมรรถนะของระบบคอมพิวเตอร์.....	11
2.2	เวลาตอบสนอง และ ปริมาณงานที่ทำได้ต่อหน่วยเวลา.....	13
2.3	เกณฑ์เปรียบเทียบสมรรถนะ (Benchmark).....	15
2.3.1	SPEC Benchmark Suites.....	16
2.3.2	เกณฑ์เปรียบเทียบสมรรถนะที่ดี.....	17
2.4	การเปรียบเทียบสมรรถนะ.....	18
2.5	CPU Time.....	21
2.6	กฎของ Amdahl.....	27
2.7	แนวทางการปรับปรุงสมรรถนะของระบบคอมพิวเตอร์.....	30
2.8	สรุป.....	31
2.9	แบบฝึกหัดท้ายบท.....	33
3	สถาปัตยกรรมชุดคำสั่ง.....	35
3.1	สถาปัตยกรรมชุดคำสั่งในอดีต.....	37
3.1.1	สถาปัตยกรรมแบบ Accumulator.....	39
3.1.2	สถาปัตยกรรมแบบ Stack.....	40
3.1.3	สถาปัตยกรรมแบบ Memory-Memory.....	42
3.1.4	สถาปัตยกรรมแบบ Register-Memory.....	43
3.1.5	สถาปัตยกรรมแบบ Register-Register.....	45
3.2	หมวดคำสั่งและชุดคำสั่ง.....	46
3.3	การจัดการหน่วยความจำ (Memory Organization).....	49
3.3.1	การจัดลำดับข้อมูล (Endian).....	50
3.3.2	การกำหนดเลขที่อยู่เริ่มต้นของข้อมูล (Memory Alignment).....	51
3.4	การอ้างอิงเลขที่อยู่หน่วยความจำ (Addressing Mode).....	54
3.5	รูปแบบคำสั่ง (Instruction Format).....	56
3.6	สถาปัตยกรรมชุดคำสั่งที่ดี.....	58
3.6.1	บทบาทของคอมพิวเตอร์ในการหาค่าเหมาะสมที่สุด.....	59

3.7	การเรียกใช้โปรแกรมย่อย และ Exception.....	62
3.7.1	การส่งผ่านและคืนค่าระหว่างโปรแกรมย่อย.....	62
3.7.2	การบันทึกค่าตัวแปรท้องถิ่นและเรจิสเตอร์.....	63
3.7.3	Exception และ Interrupt.....	64
3.8	สรุป.....	65
3.9	แบบฝึกหัดท้ายบท.....	67
4	การออกแบบ หน่วยประมวลผลกลาง แบบ single cycle.....	69
4.1	ขั้นตอนในการออกแบบหน่วยประมวลผลกลาง.....	70
4.1.1	สถาปัตยกรรมชุดคำสั่ง nanoLADA.....	71
4.2	องค์ประกอบภายในสำหรับสถาปัตยกรรม nanoLADA.....	72
4.2.1	ชุดเรจิสเตอร์ (Register file).....	73
4.2.2	Extender.....	75
4.2.3	หน่วยความจำ.....	76
4.3	ทางเดินข้อมูลสำหรับสถาปัตยกรรมชุดคำสั่ง nanoLADA.....	77
4.3.1	ทางเดินข้อมูลสำหรับคำสั่ง ORI และ ORUI.....	78
4.3.2	ทางเดินข้อมูลสำหรับคำสั่ง ADD.....	79
4.3.3	ทางเดินข้อมูลสำหรับคำสั่ง LW.....	81
4.3.4	ทางเดินข้อมูลสำหรับคำสั่ง SW.....	81
4.3.5	ทางเดินข้อมูลสำหรับคำสั่ง BEQ.....	82
4.3.6	ทางเดินข้อมูลของคำสั่ง JMP.....	83
4.3.7	สรุปทางเดินข้อมูลสำหรับสถาปัตยกรรมชุดคำสั่ง nanoLADA.....	84
4.4	สัญญาณควบคุม (Control Signals).....	86
4.5	วิธีวิกฤตและความเร็วสัญญาณนาฬิกา.....	91
4.6	แบบฝึกหัดท้ายบท.....	93
5	หน่วยประมวลผลกลางแบบ multiple cycle.....	95
5.1	สมรรถนะของหน่วยประมวลผลกลางแบบ multiple cycle.....	98
5.2	ทางเดินข้อมูลสำหรับสถาปัตยกรรม nanoLADA แบบ multiple cycle.....	100
5.3	สัญญาณควบคุมสำหรับ multiple cycle processor.....	105
5.4	ไมโครโปรแกรม (Microprogram).....	108
5.4.1	Microsequencer.....	108
5.5	แบบฝึกหัดท้ายบท.....	113
6	การเพิ่มสมรรถนะด้วย Pipeline.....	115
6.1	สมรรถนะของ Pipeline กรณีอุดมคติ.....	116
6.2	ปัญหาที่เป็นอุปสรรคต่อการทำ Pipeline.....	118
6.2.1	Structural Hazard.....	119
6.3	Data Hazard.....	122
6.3.1	การแก้ปัญหา Data Hazard ด้วยวิธีการทางซอฟต์แวร์.....	123

6.3.2	การแก้ปัญหา Data Hazard ด้วยวิธีการ Hardware Forward.....	125
6.3.3	Data Hazard แบบ RAW, WAW, WAR.....	128
6.4	Control Hazard หรือ Branch Hazard.....	130
6.4.1	การแก้ปัญหา Control Hazard ด้วยวิธีการ Branch Delay Slot.....	131
6.5	Data Path.....	133
6.6	สรุป Pipeline.....	135
6.6.1	สมรรถนะของ Pipeline ในทางปฏิบัติ.....	135
6.7	แบบฝึกหัดท้ายบท.....	137
7	สถาปัตยกรรมร่วมสมัย.....	139
7.1	อนุกรมวิธานแบบ Flynn (Flynn's Taxonomy).....	139
7.1.1	Single Instruction Stream Single Data Stream (SISD).....	139
7.1.2	Single Instruction Stream Multiple Data Stream (SIMD).....	140
7.1.3	Multiple Instruction Stream Single Data Stream (MISD).....	140
7.1.4	Multiple Instruction Stream Multiple Data Stream (MIMD).....	141
7.2	สถาปัตยกรรมแบบเวกเตอร์.....	141
7.3	สถาปัตยกรรมแบบ SuperScalar.....	143
7.4	สถาปัตยกรรมแบบ Very Long Instruction Word (VLIW).....	144
7.5	การสนับสนุนของตัวแปลภาษาในสถาปัตยกรรมร่วมสมัย.....	145
7.5.1	Loop unroll.....	146
7.5.2	Software pipeline.....	148
7.5.3	โครงการ LLVM.....	149
7.6	สรุปสถาปัตยกรรมร่วมสมัย.....	151
7.7	แบบฝึกหัดท้ายบท.....	153
8	การจัดการหน่วยความจำ.....	155
8.1	Cache.....	156
8.2	หลักการท้องถิ่น (Locality).....	158
8.3	การจัดการ Cache เบื้องต้น.....	160
8.3.1	Direct Mapped Cache.....	161
8.4	สมรรถนะของ Cache.....	164
8.5	การลด Miss Ratio.....	167
8.6	การลด Miss Penalty.....	169
8.7	Block Size.....	171
8.8	Associativity.....	173
8.9	Replacement Algorithm.....	177
8.10	การจัดการ Cache กรณีการเขียนข้อมูล (Write Management).....	178
8.11	โปรแกรมและสมรรถนะของ Cache.....	180
8.11.1	การปรับโปรแกรมเพื่อเพิ่ม Hit.....	180

8.11.2 การปรับโปรแกรมเพื่อลด Conflict Miss.....	181
8.12 หน่วยความจำเสมือน (Virtual Memory).....	182
8.12.1 การทำงานของหน่วยความจำเสมือน.....	183
8.13 การทำงานร่วมกันของ Cache และหน่วยความจำเสมือน.....	188
8.14 การเลื่อนขั้นตอนของ Cache และหน่วยความจำเสมือน.....	188
8.15 สรุป.....	190
8.16 แบบฝึกหัดท้ายบท.....	191
ภาคผนวก.....	193
ภาคผนวก ก สถาปัตยกรรมชุดคำสั่ง nanoLADA.....	195
ก.1 รูปแบบคำสั่ง 3 รูปแบบ.....	195
ก.2 คำสั่งของสถาปัตยกรรม nanoLADA.....	195
ภาคผนวก ข Verilog HDL ของสถาปัตยกรรม nanoLADA.....	197
ข.1 Register Files.....	197
ข.2 Extender.....	198
ข.3 ALU.....	198
ข.4 ADDER.....	199
ข.5 Mux 2:1.....	200
ข.6 control unit.....	200
ข.7 nanoCPU.....	203
ข.8 Micro PC.....	208
ภาคผนวก ค กิจกรรมเพิ่มเติม.....	209
ค.1 กิจกรรมบทบาทของคอมพิวเตอร์.....	209
ค.2 กิจกรรมโครงสร้าง Cache และสมรรถนะ.....	213
บรรณานุกรม.....	215

ดรรรชนีรูปภาพ

รูปที่ 1.1: ส่วนประกอบของระบบคอมพิวเตอร์.....	3
รูปที่ 1.2: ตัวอย่างขั้นตอนการแปลผลโปรแกรม.....	5
รูปที่ 3.1: ทางเดินข้อมูลของสถาปัตยกรรมแบบ accumulator.....	39
รูปที่ 3.2: ทางเดินข้อมูลของสถาปัตยกรรมแบบ stack.....	41
รูปที่ 3.3: ทางเดินข้อมูลของสถาปัตยกรรมแบบ memory-memory.....	42
รูปที่ 3.4: ทางเดินข้อมูลของสถาปัตยกรรมแบบ register-memory.....	44
รูปที่ 3.5: โครงสร้างทางเดินข้อมูลของสถาปัตยกรรมแบบ register-register.....	45
รูปที่ 3.6: การเรียงข้อมูล (a) big endian (b) little endian.....	50
รูปที่ 3.7: การต่อหน่วยความจำเข้ากับหน่วยประมวลผล (a) ความกว้างสายสัญญาณเป็นหนึ่ง (b) ความกว้างสายสัญญาณเป็นสอง.....	52
รูปที่ 3.8: การอ่านค่าแบบมี restricted alignment และ แบบ unrestricted alignment.....	53
รูปที่ 3.9: โครงสร้างการจัดวางหน่วยความจำ (word addressing แสดงในเลขฐาน 16) (a) โปรแกรมภาษาซี (b) สถาปัตยกรรมแบบ unrestricted alignment และ (c) สถาปัตยกรรมแบบ restricted alignment.....	54
รูปที่ 3.10: รูปแบบคำสั่ง (instruction format) ในสถาปัตยกรรม nanoLADA.....	58
รูปที่ 3.11: เปรียบเทียบการทำงานระหว่าง caller save และ callee save.....	64
รูปที่ 4.1: ความสัมพันธ์ระหว่างสัญญาณนาฬิกา วงจรเชิงผสม และ วงจรเชิงลำดับ.....	70
รูปที่ 4.2: ชุดเรจิสเตอร์ขนาด 32x32 บิต สามารถอ่านได้ 2 ชุด และเขียนได้ 1 ชุดพร้อมกัน.....	74
รูปที่ 4.3: การทำงานของ Extender (a) zero extender (b) sign extender (c) zero padding.....	75
รูปที่ 4.4: โครงสร้างหน่วยความจำ.....	76
รูปที่ 4.5: ทางเดินข้อมูลสำหรับ $PC \leftarrow PC+4$	78
รูปที่ 4.6: ทางเดินข้อมูลเมื่อเพิ่มคำสั่ง ORI และ ORUI.....	79
รูปที่ 4.7: ทางเดินข้อมูลเมื่อเพิ่มคำสั่ง ADD.....	80
รูปที่ 4.8: ทางเดินข้อมูลเมื่อเพิ่มคำสั่ง LW.....	81
รูปที่ 4.9: ทางเดินข้อมูลเมื่อเพิ่มคำสั่ง SW.....	82
รูปที่ 4.10: ทางเดินข้อมูลส่วน PC เมื่อเพิ่มคำสั่ง BEQ.....	83
รูปที่ 4.11: ทางเดินข้อมูลส่วน PC เมื่อเพิ่มคำสั่ง JMP.....	84
รูปที่ 4.12: ทางเดินข้อมูลสำหรับสถาปัตยกรรม nanoLADA.....	85
รูปที่ 5.1: เปรียบเทียบเวลา (a) คำสั่งที่ไม่ต้องอ่านข้อมูลจากหน่วยความจำ และ (b) คำสั่งที่ต้องอ่านข้อมูลจากหน่วยความจำ (กำหนดให้ความกว้างแทนเวลา).....	95
รูปที่ 5.2: ตัวอย่างการแบ่งขั้นตอนการทำงาน (a) การประมวลผลแบบ single cycle (b) การประมวลผลแบบ multiple cycle.....	96
รูปที่ 5.3: การแบ่งขั้นตอนการทำงาน.....	97

รูปที่ 5.4: การแทรกเรจิสเตอร์เงาเพื่อปรับทางเดินข้อมูลให้รองรับการทำงานแบบ multiple cycle.....	101
รูปที่ 5.5: ทางเดินข้อมูลสำหรับ multiple-cycle processor.....	105
รูปที่ 5.6: state diagram แสดงการทำงานของหน่วยประมวลผลกลางแบบ multiple cycle	106
รูปที่ 5.7: โครงสร้างของ Microsequencer.....	109
รูปที่ 5.8: state diagram สำหรับไมโครโปรแกรม.....	110
รูปที่ 6.1: เปรียบเทียบเวลาการทำงานแบบมี pipeline และ ไม่มี pipeline.....	116
รูปที่ 6.2: ปัญหา Structural Hazard.....	119
รูปที่ 6.3: ปัญหา structure hazard จากการใช้คำสั่งใช้จำนวน cycle ไม่เท่ากัน.....	120
รูปที่ 6.4: การแก้ปัญหา Structural Hazard ด้วยการ stall.....	121
รูปที่ 6.5: การแก้ structural hazard โดยการปรับขั้นตอนการทำงานของคำสั่ง R-type.....	122
รูปที่ 6.6: ปัญหา data hazard.....	123
รูปที่ 6.7: การแก้ปัญหา data hazard ด้วย hardware stall.....	126
รูปที่ 6.8: ตัวอย่างการแก้ปัญหา Data Hazard ด้วย hardware forward.....	127
รูปที่ 6.9: ตัวอย่างกรณีที่ไม่สามารถ forward ได้.....	127
รูปที่ 6.10: การทำ hardware stall กรณีที่ไม่สามารถทำ hardware forward ได้.....	128
รูปที่ 6.11: data hazard แบบ RAW, WAW และ WAR.....	129
รูปที่ 6.12: ตัวอย่างการทำงานของ Pipeline เมื่อมีการ bypass เพื่อแก้ปัญหา Control Hazard.....	131
รูปที่ 6.13: การแก้ปัญหา control hazard ด้วย branch delay slot (แบบ taken) และ (not taken).....	133
รูปที่ 6.14: การแยกเรจิสเตอร์คำสั่งแยกออกจากกันในแต่ละขั้นของ pipeline.....	134
รูปที่ 6.15: การส่งผ่านสัญญาณควบคุมสำหรับ pipeline.....	134
รูปที่ 7.1: เปรียบเทียบโปรแกรมบนสถาปัตยกรรมแบบหน่วยประมวลผลปกติ และ สถาปัตยกรรมแบบเวกเตอร์.....	142
รูปที่ 7.2: สถานการณ์จำลองการทำงานแบบ SuperScalar.....	144
รูปที่ 7.3: สถานการณ์จำลองการทำงานแบบ Very Long Instruction Word.....	145
รูปที่ 7.4: การทำ loop unroll จำนวน 4 รอบ.....	146
รูปที่ 7.5: การทำ loop unroll ในภาษาแอสเซมบลี.....	147
รูปที่ 7.6: แนวคิดการทำ software pipeline.....	148
รูปที่ 7.7: การทำ software pipeline.....	149
รูปที่ 7.8: ตัวอย่าง LLVM IR.....	150
รูปที่ 8.1: ความเร็วและปริมาณหน่วยจัดเก็บข้อมูลที่มีในระบบคอมพิวเตอร์.....	157
รูปที่ 8.2: ลำดับการเข้าถึงข้อมูล.....	157
รูปที่ 8.3: ความถี่การอ้างอิงช่วงเลขที่อยู่ของ gcc.....	159
รูปที่ 8.4: ตัวอย่างการจัดคนเข้านั่งเก้าอี้ในห้องแบบ direct mapped cache.....	162

รูปที่ 8.5: โครงสร้าง direct mapped cache.....	163
รูปที่ 8.6: โครงสร้าง direct mapped cache ที่รองรับการทำงานของหลักการท้องถิ่นเชิงพื้นที่	164
รูปที่ 8.7: กรณีมี cache เพียง 1 บรรทัด.....	168
รูปที่ 8.8: การทำ multilevel cache เพื่อลด miss penalty.....	170
รูปที่ 8.9: block size ต่อ miss penalty.....	172
รูปที่ 8.10: กราฟความสัมพันธ์ระหว่าง block size, miss rate และ miss penalty.....	173
รูปที่ 8.11: ตัวอย่างการเกิด conflict miss ใน 1-way set associative cache.....	174
รูปที่ 8.12: การเพิ่ม set associative เพื่อลดปัญหา conflict miss.....	175
รูปที่ 8.13: โครงสร้างของระบบ cache แบบ 4-way set associative.....	176
รูปที่ 8.14: การใช้ Write Buffer เพื่อลดเวลาในการเขียนข้อมูลแบบ Write Through.....	179
รูปที่ 8.15: การเพิ่ม level ของ cache เพื่อบรรเทาปัญหา saturate ของ write buffer.....	179
รูปที่ 8.16: การทำงานของหน่วยความจำเสมือน.....	185
รูปที่ 8.17: การแปลง virtual address เป็น physical address.....	186
รูปที่ 8.18: การทำงานของ TLB.....	187
รูปที่ 8.19: การเข้าถึงข้อมูลในหน่วยความจำเมื่อมี cache และหน่วยความจำเสมือน.....	188
รูปที่ 8.20: การแปลง virtual address และการอ้างอิงเลขที่อยู่ของ cache.....	189
รูปที่ 8.21: โครงสร้างเลขที่อยู่ที่ไม่สามารถทำการห่อหุ้มเวลาได้.....	190

ดรรชนีตาราง

ตารางที่ 2.1:	เปรียบเทียบเวลาได้จากเกณฑ์เปรียบเทียบสมรรถนะ.....	18
ตารางที่ 2.2:	การใช้ค่าเฉลี่ยเลขคณิตถ่วงน้ำหนักเปรียบเทียบเวลา.....	19
ตารางที่ 2.3:	การเปรียบเทียบสมรรถนะด้วยอัตราส่วน พร้อมค่าเฉลี่ยเลขคณิต.....	20
ตารางที่ 2.4:	การเปรียบเทียบสมรรถนะด้วยอัตราส่วน พร้อมค่าเฉลี่ยเรขาคณิต.....	20
ตารางที่ 3.1:	การอ้างอิงเลขที่อยู่แบบต่าง ๆ และการเทียบเคียงกับภาษาระดับสูง.....	55
ตารางที่ 4.1:	การทำงานของ ALU สำหรับ nanoLADA.....	87
ตารางที่ 4.2:	สัญญาณควบคุมสำหรับสถาปัตยกรรม nanoLADA.....	89
ตารางที่ 5.1:	สัญญาณควบคุมสำหรับหน่วยประมวลผลแบบ multiple cycle.....	106
ตารางที่ 5.2:	ไมโครอินสตรักชันของสัญญาณควบคุมบางส่วน.....	110
ตารางที่ 8.1:	สมรรถนะของ replacement algorithm แบบ LRU และ RR.....	178

ดรรรชนีสมการ

สมการที่ 2.1	สมรรถนะและเวลา.....	11
สมการที่ 2.2	การหา Speedup จากสมรรถนะ.....	12
สมการที่ 2.3	การหา Speedup จากเวลา.....	12
สมการที่ 2.4	Elapse Time.....	15
สมการที่ 2.5	Cycle Time.....	22
สมการที่ 2.6	average CPI.....	22
สมการที่ 2.7	CPU Time/Instruction.....	23
สมการที่ 2.8	Seconds/Instruction.....	24
สมการที่ 2.9	CPU Time เมื่อทราบเวลาต่อคำสั่ง.....	24
สมการที่ 2.10	CPU Time เมื่อทราบ Cycle ต่อคำสั่ง.....	24
สมการที่ 2.11	Execution Time.....	28
สมการที่ 2.12	Overall Speedup.....	29
สมการที่ 6.1	CPU TIME (pipeline).....	118
สมการที่ 6.2	สมรรถนะของ pipeline เมื่อมี stall cycle.....	135
สมการที่ 8.1	Memory Access Time (หน่วย cycle).....	166
สมการที่ 8.2	ค่า CPI กรณีที่มี cache.....	166

1 บทนำ

1.1 รู้ก่อนเรียน

เพื่อให้การศึกษาวិชาสาสถาปัตยกรรมคอมพิวเตอร์ง่ายขึ้น ผู้เรียนควรมีทักษะในการออกแบบวงจรตรรกะ หรือวงจรดิจิทัล โดยองค์ความรู้ทางวงจรดิจิทัลที่มีนั้น ควรจะเพียงพอที่สามารถออกแบบได้ทั้งวงจรแบบผสม (combinational logic circuit) และวงจรเชิงลำดับ (sequential circuit) นอกจากนี้ผู้เรียนควรมีความรู้พื้นฐานทางซอฟต์แวร์พอสมควรเพื่อประกอบความเข้าใจ กล่าวคือ ผู้เรียนควรมีประสบการณ์ในการเขียนโปรแกรม (ไม่จำกัดภาษา) ทั้งนี้หากผู้เรียนมีประสบการณ์กับภาษาแอสเซมบลี (assembly) ไม่ว่าจะเป็นสถาปัตยกรรมใด จะช่วยให้สามารถเข้าใจเนื้อหาบางส่วนได้ง่ายและรวดเร็วยิ่งขึ้นเป็นพิเศษ แต่หากไม่มีความคุ้นเคยกับภาษาแอสเซมบลี อย่างน้อยผู้เรียนควรจะเข้าใจหลักการเบื้องต้น เช่น ตัวแปร การจองหน่วยความจำ ประโยคเงื่อนไขและการเรียกใช้งานโปรแกรมย่อย ในกรณีของพื้นฐานความรู้ด้านการเขียนโปรแกรม ปัจจุบันไม่น่าจะเป็นประเด็นแต่อย่างไร เนื่องจากการจากเรียนการสอนทางด้านวิทยาศาสตร์และวิศวกรรมศาสตร์สมัยใหม่ มักสอดแทรกองค์ความรู้เกี่ยวกับการเขียนโปรแกรมเข้าไปอยู่แล้ว เช่น นิสิตนักศึกษาชั้นปีที่หนึ่งในหลายหลักสูตรทางด้านคอมพิวเตอร์ มักจะได้เรียนวิชาการทำโปรแกรมด้วยภาษาจาวา (Java programming) หรือ การทำโปรแกรมด้วยภาษาไพธอน (Python programming) และนิสิตนักศึกษาชั้นปีที่สองทางสายวิศวกรรมคอมพิวเตอร์ มักจะได้เรียนภาษาซี (C programming) และพื้นฐานการออกแบบวงจรตรรกะหรือวงจรดิจิทัล

เนื้อหาในหนังสือเล่มนี้เรียบเรียงมาจากตำราต่าง ๆ ทั้งฉบับคลาสสิก (ตำนาน) ฉบับอ้างอิง ฉบับอ่านเล่น รวมถึงงานตีพิมพ์และงานวิจัยด้านต่าง ๆ เนื่องจากสถาปัตยกรรมคอมพิวเตอร์เป็นศาสตร์ที่มีการพัฒนา ปรับปรุงและเปลี่ยนแปลงอยู่เสมอ ผู้เรียนควรค้นคว้าเพิ่มเติมจากแหล่งความรู้อื่นประกอบด้วย

โครงสร้างและสถาปัตยกรรมคอมพิวเตอร์ที่อธิบายในที่นี้ จะอธิบายโดยอาศัยพื้นฐานจากสถาปัตยกรรมชุดคำสั่งสมมติ ชื่อ nanoLADA ซึ่งเป็นฉบับย่อของ LADA (Light-weighted Architecture for Design and Analysis) เป็นหลัก โดย LADA เป็นสถาปัตยกรรมคอมพิวเตอร์ประเภท RISC แบบ 32 บิต ที่ผู้เขียนพัฒนาขึ้นเพื่อประกอบการเรียนการสอนโดยเฉพาะ (ผู้เรียนยังไม่ต้องกังวลว่า สถาปัตยกรรมคอมพิวเตอร์ประเภท RISC คืออะไร เมื่อถึงเนื้อหาส่วนที่เกี่ยวข้องก็จะเข้าใจเอง) สถาปัตยกรรม LADA มุ่งเน้นให้มิโครสร้างเข้าใจได้ง่าย ไม่ซับซ้อน เหมาะกับการศึกษาในระดับเบื้องต้น ในส่วนที่

เป็นการกล่าวถึงเรื่องต่อยอดต่าง ๆ อาจมีการแทรกเนื้อหาของสถาปัตยกรรมอื่นเพื่อประกอบการอธิบายเพิ่มเติมบ้าง การแทรกรายละเอียดของสถาปัตยกรรมเหล่านี้ เน้นเพื่อให้เกิดการเปรียบเทียบและเห็นข้อแตกต่าง ทั้งนี้ผู้เขียนจะไม่แทรกเนื้อหาให้มากเกินไป โดยการสอดแทรกจะขึ้นกับความต่อเนื่องและความเหมาะสมของเนื้อหาเป็นหลัก ในตัวอย่างประกอบคำอธิบาย จะพยายามใช้สถาปัตยกรรม LADA เป็นพื้นฐาน (ยกเว้นกรณีที่ต้องการเปรียบเทียบกับสถาปัตยกรรมอื่น) เพื่อให้เกิดความต่อเนื่องของเนื้อหา

1.2 ส่วนประกอบของระบบคอมพิวเตอร์

ระบบคอมพิวเตอร์โดยทั่วไป ประกอบด้วยหน่วยประมวลผลกลาง แป้นพิมพ์ เมาส์ จอภาพ เครื่องพิมพ์ ระบบสื่อประสม (multimedia) และอุปกรณ์ต่อพ่วงอีกมากมาย โครงสร้างดังกล่าวสามารถจำแนกตามการทำงานได้เป็น

- หน่วยประมวลผลกลาง (central processing unit)
- อินพุต (keyboard, mouse, etc...)
- เอาท์พุต (display, printer, etc...)
- หน่วยความจำ (memory)
- หน่วยจัดเก็บข้อมูล (storage, disk, cd, tape, etc)

การแบ่งในลักษณะนี้ เป็นการแบ่งส่วนประกอบของคอมพิวเตอร์ตามมุมมองของผู้ใช้งานทั่วไป ในทางสถาปัตยกรรมนิยมแบ่งส่วนประกอบของคอมพิวเตอร์เป็นสามส่วนหลักตามลักษณะของหน้าที่ดังแสดงในรูปที่ 1.1 ได้แก่ หน่วยประมวลผลกลาง หน่วยความจำ และระบบอินพุตเอาท์พุต จะสังเกตเห็นว่า อินพุต เอาท์พุต และ หน่วยจัดเก็บข้อมูล ถูกมองรวมเป็นระบบเดียวกันทั้งหมด ส่วนหนึ่งเนื่องจากในทางสถาปัตยกรรม อุปกรณ์เหล่านี้เชื่อมต่อกับหน่วยประมวลผลกลางผ่านระบบสายสัญญาณย่อยอินพุต/เอาต์พุต (I/O subsystem) เหมือนกัน ต่างจากหน่วยความจำที่มักจะมีสายสัญญาณเชื่อมต่อเฉพาะสำหรับหน่วยความจำ (memory bus)

หนังสือเล่มนี้มุ่งความสนใจไปที่หน่วยประมวลผลกลางเป็นหลักก่อน จากนั้นจึงกล่าวถึงองค์ประกอบรอบข้าง เช่น หน่วยความจำและส่วนต่อขยายต่าง ๆ เมื่อส่วนดังกล่าวมีผลกับสมรรถนะของระบบ จนส่งผลให้การออกแบบและการวิเคราะห์แตกต่างไป