บทที่ 3
โพรเซสเซอร์และการทำงาน

หน้าที่หลักของไมโครโพรเซสเซอร์คือการคำนวณทางด้านคณิตศาสตร์และลอจิก โดยไมโครโพรเซสเซอร์จะทำการอ่านโปรแกรมจากหน่วยความจำโดยไมโครโพรเซสเซอร์จะเชื่อมต่อกับหน่วยความจำและระบบอินพุตเอาต์พุตผ่านกลุ่มของสายสัญญาณที่เราเรียกว่า bus โดยบัสของไมโครโพรเซสเซอร์จะประกอบไปด้วย สัญญาณ 3 กลุ่มดังแสดงในสไลด์หน้าที่ 4 ได้แก่

- Address bus
- Data bus
- Control bus

สัญญาณ Address bus ใช้ในการระบุหมายเลขตำแหน่งของหน่วยความจำ โดยจำนวนบิตของสัญญาณ Address bus จะเป็นตัวบอกว่าไมโครโพรเซสเซอร์สามารถอ้างหน่วยความจำสูงสุดได้เท่าใด ไมโครโพรเซสเซอร์ที่มีจำนวน Address bus จำนวน n เส้นจะสามารถอ้างหน่วยความจำได้สูงสุดเท่ากับ 2^n แอดเดรส ส่วนสัญญาณ Data bus ใช้ในการรับส่งข้อมูลระหว่างไมโครโพรเซสเซอร์กับหน่วยความจำและระบบอินพุตเอาต์พุต โดยจำนวนบิตของสัญญาณ Data bus จะบอกถึงความสามารถในการอ่านหน่วยความจำว่าการอ่านหน่วยความจำ 1 ครั้งสามารถได้ข้อมูลจำนวนกี่บิต ส่วนสัญญาณ Control bus จะใช้ระบุว่าในการติดต่อกับหน่วยความจำหรืออุปกรณ์ I/O นั้นต้องการทำโอเปอเรชั่นอะไร เช่น การอ่านหน่วยความจำ การเขียนหน่วยความจำ การอ่านอุปกรณ์ I/O การเขียนข้อมูลลงสู่อุปกรณ์ I/O เป็นต้น

ในการระบุขนาดของไมโครโพรเซสเซอร์จะดูที่ความสามารถในการคำนวณตัวเลขในแต่ละครั้งของไมโครโพรเซสเซอร์ ยกตัวอย่างเช่น ซีพียู 32 บิต จะหมายถึงซีพียูตัวนั้นสามารถทำโอเปอเรชั่นกับตัวเลขขนาด 32 บิตได้โดยอาศัยการประมวลผลคำสั่งเพียงครั้งเดียว ความจริงซีพียูขนาด 8 บิตก็สามารถที่จะคำนวณตัวเลขขนาด 32 บิตได้ แต่ต้องทำการคำนวณหลายๆ ครั้งซึ่งผิดกับซีพียูขนาด 32 บิตที่สามารถคำนวณเสร็จได้ใน 1 ครั้ง จะเห็นว่าการเพิ่มขนาดของไมโครโพรเซสเซอร์จะช่วยลดจำนวนครั้งในการประมวลผลตัวเลขขนาดใหญ่ๆ ได้ ส่งผลให้ความเร็วในการทำงานเพิ่มขึ้น ในอดีตเราเคยเข้าใจกันผิดๆ ว่าขนาดของไมโครโพรเซสเซอร์ดูที่ขนาดข้อมูลที่ซีพียูสามารถอ่านได้ในแต่ละครั้ง ซึ่งไม่เป็นความจริง ยกตัวอย่างเช่น ซีพียู Pentium ซึ่งมีขนาด Data bus ขนาด 64 บิต แต่สามารถคำนวณตัวเลขได้ครั้งละ 32 บิตเท่านั้น ตารางที่ 3.1 แสดงขนาดของไมโครโพรเซสเซอร์และขนาดของแอดเดรสบัสและดาต้าบัส

ตารางที่ 3.1 ขนาดของไมโครโพรเซสเซอร์และแอดเดรสบัสและดาต้าบัส

ซีพียู	ขนาดของซีพียู	ขนาดของ Address bus	ขนาดของ Databus
8088	16	20	8
8051	8	16	8
80286	16	24	16
80386DX	32	32	32
80386SX	32	32	16
Pentium	32	32	64

สไลด์หน้าที่ 8 แสดงการทำงานของซีพียูซึ่งจะประกอบไปด้วยรอบการทำงาน 2 ขั้นตอนคือ Fetch และ Execute โดยที่ การเฟตช์ (Fetch) หมายถึงการอ่านคำสั่งจากหน่วยความจำเข้ามาใส่ในตัวโพรเซสเซอร์ ส่วนการเอกซีคิวต์ (Execute) จะหมายถึงการปฏิบัติการตามที่โอเปอเรชั่นของคำสั่งระบุจนเสร็จสิ้น เราเรียก 2 ขั้นตอนนี้ว่า Instruction cycle

สไลด์หน้าที่ 9-10 แสดงความสัมพันธ์ระหว่าง Stored program concept กับระบบบัสของไมโครโพรเซสเซอร์ โดยซีพียูจะเชื่อมต่อกับหน่วยความจำผ่านบัสทั้งสามบัส โดยต่อบัสแอดเดรสออกจากรีจิสเตอร์ MAR (Memory Address Register) ไปยังหน่วยความจำ และซีพียูรับคำสั่งจากหน่วย ความจำผ่านดาต้าบัสไปเก็บไว้ในรีจิสเตอร์ IR (Instruction Register)

สไลด์หน้าที่ 12 กล่าวถึงประเภทของคำสั่งของไมโครโพรเซสเซอร์ซึ่งสามารถแบ่งออกได้เป็น 4 จำพวกใหญ่ๆ ได้แก่

1. คำสั่งเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำและรีจิสเตอร์ของซีพียู
2. คำสั่งคำนวณทางด้าน Arithmetic และ Logic บนข้อมูลที่ได้รับมา
3. คำสั่งควบคุมการทำงานของโปรแกรม (Program Sequencing and Control)
4. คำสั่งจัดการอุปกรณ์อินพุตเอาต์พุต (I/O transfer)

คำสั่งของซีพียูประกอบด้วย 2 ส่วน คือ โอเปอเรเตอร์ และโอเปอแรนด์ โดยโอเปอเรเตอร์ จะระบุว่าให้คำสั่งทำอะไร ส่วนโอเปอแรนด์จะระบุว่าคำสั่งนั้น ทำกับข้อมูลซึ่งเก็บอยู่ในไหน เนื่องจากคำสั่งของซีพียูมีหลายแบบ คำสั่งของซีพียูบางสถาปัตยกรรมสามารถระบุโอเปอแรนด์ได้ถึง 3 ตัวในคำสั่ง ในขณะที่บางสถาปัตยกรรมสามารถระบุจำนวนโอเปอแรนด์ในคำสั่งได้เพียงแค่ตัวเดียว โดยในสไลด์หน้าที่ 13 แสดงคำสั่งของซีพียูแบบ 3-address ซึ่งสามารถระบุโอเปอแรนด์ได้ 3 ตัว ยกตัวอย่างเช่น คำสั่ง ADD R2, R0, R1ซึ่งมีโอเปอเรเตอร์คือ ADD ซึ่งเป็นการสั่งให้ทำการบวก ส่วนโอเปอแรนด์คือ ข้อมูลใน R0, R1 และเก็บผลลัพธ์ไว้ในรีจิสเตอร์ R2

โปรแกรมในสไลด์หน้าที่ 13 เป็นการเขียนคำสั่งของซีพียูที่มีคำสั่งแบบ 3-address ในการบวกค่าในตัวแปร A, B, C เข้าด้วยกันและเก็บผลลัพธ์ในตัวแปร D จะเห็นว่าต้องใช้คำสั่งของซีพียูถึง 6 คำสั่งเพื่อทำโปรแกรมดังกล่าว

โปรแกรมในสไลด์หน้าที่ 14 เป็นการเขียนโปรแกรมหน้าที่เดียวกับโปรแกรมในสไลด์หน้าที่ 13 เพียงแต่เปลี่ยนมาใช้ซีพียูที่มีคำสั่งแบบ 2-address จะเห็นว่าแม้ซีพียูมีจำนวนโอเปอแรนด์ของคำสั่งน้อยกว่าก็สามารถทำงานได้เช่นเดียวกับซีพียูที่มีคำสั่งแบบ 3-address

โปรแกรมในสไลด์หน้าที่ 15 แสดงการเขียนโปรแกรมหน้าที่เดียวกับโปรแกรมหน้าที่ 13 และ 14 แต่ใช้ซีพียูที่มีคำสั่งแบบ 1-address ซึ่งจะเห็นว่าใช้คำสั่งเพียง 4 คำสั่ง โดยในสถาปัตยกรรมแบบนี้จะมีรีจิสเตอร์ภายในซีพียูที่เรียกว่า Accumulator(ACC) ซึ่งทุกคำสั่งจะทำงานกับข้อมูลในแอคคิวมูเลเตอร์และตำแหน่งหน่วยความจำที่ระบุ ยกตัวอย่างเช่น คำสั่ง LOAD [10000] หมายถึงให้ทำการอ่านข้อมูลในหน่วยความจำตำแหน่งที่ 10000 มาใส่ในแอคคิวมูเลเตอร์ และคำสั่ง ADD [10001] ซึ่งจะทำการบวกค่าในแอคคิวมูเลเตอร์กับหน่วยความจำตำแหน่งที่ 10001 เข้าด้วยกัน และเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์ ส่วนคำสั่ง STORE [10003] จะเป็นการนำข้อมูลในแอคคิวมูเลเตอร์มาเก็บในหน่วยความจำตำแหน่ง 10003

รีจิสเตอร์ของซีพียูส่วนใหญ่แบ่งเป็น 2 ชนิด ได้แก่ รีจิสเตอร์ใช้งานทั่วไปและรีจิสเตอร์ใช้งานเฉพาะหน้าที่ดังแสดงในสไลด์หน้าที่ 16 โดยรีจิสเตอร์ใช้งานพิเศษที่สำคัญคือ รีจิสเตอร์ PC (ProgramCounter) ซึ่งทำหน้าที่ชี้ตำแหน่งของหน่วยความจำที่ซีพียูจะเฟตช์คำสั่งเข้ามาทำงานดังแสดงในสไลด์หน้าที่ 18-23 โดยปกติค่ารีจิสเตอร์ PC จะเพิ่มค่าขึ้นทีละเป็นจำนวนไบต์ของคำสั่งยกเว้นมีการบรานช์ของโปรแกรมดังแสดงในสไลด์หน้าที่ 24 และอีกรีจิสเตอร์ที่สำคัญซึ่งแสดงให้เห็น Flag หรือ Condition code ซึ่งแสดงสถานะการทำงานภายหลังจากการทำคำสั่งทางคณิตศาสตร์และลอจิก ในสไลด์หน้าที่ 25 แสดงแฟลกของซีพียู Z80 ซึ่งมีค่าแฟลกที่สำคัญคือ Carry flag ซึ่งบอกว่ามีการทดหรือยืมในการทำคำสั่งบวกหรือลบ ตามลำดับ ส่วนแฟลก Overflow ใช้ในการตรวจเช็คว่าค่าผลลัพธ์จากการทำคำสั่งทางคณิตศาสตร์แล้วมีการเกินของค่าจากย่านที่ขนาดของซีพียูจะเก็บได้ซึ่งได้เรียนมาแล้วจากบทที่ 2 ในเรื่องการตรวจเช็คสถานะ Overflow

ส่วนประกอบหลักของซีพียูในมุมมองของผู้ใช้โดยทั่วไปจะประกอบไปด้วยส่วนหลัก 2 ส่วนคือ ALU และ CU แต่ในมุมมองของผู้พัฒนาซีพียูจะมองว่าซีพียูประกอบไปด้วย 2 ส่วนดังแสดงในสไลด์หน้าที่ 45 คือ Datapath และ Control Unit ส่วน Datapath จะประกอบไปด้วยรีจิสเตอร์ต่างๆ และเส้นทางการเดินข้อมูลรวมทั้งตัว ALU ด้วย ดังนั้นจึงสามารถมองได้ว่า ALU เป็นสับเซ็ทของส่วน Datapath นั่นเอง

สไลด์หน้าที่ 46 แสดงโครงสร้าง Datapath ภายในของซีพียูซึ่งมีโครงสร้างบัสภายในแบบบัสเดี่ยว โดยประกอบไปด้วยรีจิสเตอร์ใช้งานทั่วไปซึ่งมองเห็นได้โดยโปรแกรมเมอร์ คือได้แก่รีจิสเตอร์ R0-R(n-1) และนอกจากนี้ยังมีรีจิสเตอร์และอุปกรณ์ที่โปรแกรมเมอร์มองไม่เห็นอีกหลายตัว อันได้แก่

1. รีจิสเตอร์ PC (Program Counter) ทำหน้าที่เก็บแอดเดรสของคำสั่งที่ซีพียูจะทำการเฟตช์คำสั่งเข้ามาทำงาน
2. รีจิสเตอร์ MAR (Memory Address Register) ทำหน้าที่เก็บค่าแอดเดรสของหน่วยความจำที่ซีพียูต้องการติดต่อ
3. รีจิสเตอร์ MDR (Memory Data Register) ทำหน้าที่รับ/ส่งข้อมูลระหว่าง Data bus ภายนอกของไมโครโพรเซสเซอร์และบัสภายในของไมโครโพรเซสเซอร์
4. รีจิสเตอร์ IR (Instruction Register) ใช้ในการเก็บคำสั่งปัจจุบันที่ซีพียูอ่านได้จากหน่วยความจำเพื่อทำการถอดรหัสว่าจะทำ Operation ใดกับคำสั่งนั้นๆ
5. รีจิสเตอร์ Z ใช้ในการเก็บข้อมูลชั่วคราวในการทำคำสั่งทางคณิตศาสตร์ของ ALU
6. ALU (Arithmetic and Logical Unit) ทำหน้าที่คำนวณทางด้านคณิตศาสตร์และลอจิก
7. รีจิสเตอร์ Y ใช้ในการพักข้อมูลชั่วคราวก่อนที่จะป้อนสู่ ALU
8. MUX ทำหน้าที่เลือกว่าจะนำค่าคงที่ค่า “4” หรือค่าจากรีจิสเตอร์ Y ป้อนเข้าสู่ ALU

ในตัวอย่างนี้จะถือว่าซีพียูในสไลด์หน้าที่ 46 เป็นซีพียูขนาด 32 บิต โดยการทำงานของซีพียูจะแสดงให้เห็นในสไลด์หน้าที่ 47 โดยเริ่มจากการเฟตช์คำสั่งโดยส่งค่าจากรีจิสเตอร์ PC ไปใส่ในรีจิสเตอร์ MAR ค่าจากรีจิสเตอร์ MAR จะต่อโดยตรงกับแอดเดรสบัส ซีพียูจะแอกตีฟสัญญาณ Memory Read ที่ Control bus เพื่อบอกหน่วยความจำว่าต้องการอ่านคำสั่งจากหน่วยความจำ หน่วยความจำจะส่งข้อมูลตำแหน่งที่ระบุมาให้โพรเซสเซอร์ผ่านดาต้าบัสมาเก็บไว้ในรีจิสเตอร์ MDR จากนั้นซีพียูจะสั่งให้รีจิสเตอร์ MDR ส่งข้อมูลในรีจิสเตอร์ IR ผ่านทางบัสภายใน หลังจากที่รีจิสเตอร์ IR ได้คำสั่งเข้ามา ก็จะต้องเพิ่มค่า PC ให้ไปชี้ตำแหน่งถัดไป โดยจะต้องบวกค่า PC ด้วยค่า 4 สาเหตุที่ต้องบวกด้วยค่า 4 เพราะซีพียูขนาด 32 บิตมีขนาดของคำสั่งขนาด 32 บิตซึ่งต้องการจำนวนแอดเดรสของหน่วยความจำจำนวน 4 แอดเดรสในการเก็บ โดย 1 ตำแหน่งแอดเดรสเก็บข้อมูลได้ 8 บิตหรือ 1 ไบต์

ขั้นตอนการเอกซีคิวต์จะเริ่มจากการที่ซีพียูถอดรหัสคำสั่งว่าคำสั่งนั้นเป็นคำสั่งอะไร ยกตัวอย่าง เช่น ในสไลด์หน้าที่ 48 เป็นการเอกซีคิวต์คำสั่ง ADD R1,R2 ซึ่งเป็นการนำค่าในรีจิสเตอร์ R1, R2 มาบวกค่ากัน แล้วเก็บผลลัพธ์ลงในรีจิสเตอร์ R1 ก็จะมีการทำงานเริ่มจากการอ่านค่าจากรีจิสเตอร์ R1 ไปเก็บไว้ในรีจิสเตอร์ Y และสั่งให้มัลติเพลกเซอร์เลือกค่า Y ลงสู่ ALU จากนั้นจึงส่งค่า R2 ให้ ALU ผ่านทาง Internal bus จากนั้น Control Unit จึงสั่งให้ ALU ทำการบวกค่าใส่ในรีจิสเตอร์ Z หลังจากนั้นจึงนำค่าในรีจิสเตอร์ Z เขียนค่าลงในรีจิสเตอร์ R1 เป็นอันเสร็จสิ้นการเอกซีคิวต์คำสั่ง ADD R1,R2

จากคำสั่ง ADD R1, R2 ที่ทำงานบน Datapath แบบ Single bus ที่ผ่านมาจะเห็นว่าในขั้นตอนการเอกซีคิวต์นั้นเราไม่สามารถที่จะอ่านข้อมูลจาก R1 และ R2 ขึ้นมาพร้อมกันได้ทั้งนี้เพราะคุณลักษณะของบัสจะอนุญาตให้มีเพียงอุปกรณ์เพียงตัวเดียวเท่านั้นที่สามารถเขียนข้อมูลลงสู่บัสได้ หากเราดัดแปลงให้ Datapath มีจำนวนบัสมากขึ้นเป็น 3 บัสดังแสดงในสไลด์หน้าที่ 52 เราจะสามารถอ่านข้อมูลจากรีจิสเตอร์ R1, R2 ขึ้นมาพร้อมๆ กับเขียนผลลัพธ์ลงไปในรีจิสเตอร์ R1 ได้ ส่งผลให้เวลาในการเอกซีคิวต์คำสั่ง ADD R1,R2 ลดลง

สถาปัตยกรรมของซีพียูแบบ 3 บัส จะเหมาะมากกับซีพียูที่มีคำสั่งแบบ 3-address ในสไลด์หน้าที่ 53 แสดงให้เห็นถึงการทำงานของคำสั่ง ADD R6, R5, R4 ซึ่งเป็นแบบ 3-address โดยในสเตทที่ 1-3 จะเป็นการเฟตช์คำสั่งส่วนการเอกซีคิวต์คำสั่งจะสามารถทำได้โดยต้องการเพียงสเตท 4 เพียงสเตทเดียวในการอ่านค่าจาก R5 และ R4 เพื่อป้อนให้กับ ALU และนำผลลัพธ์จาก ALU มาเขียนลงใน R6 พร้อมๆ กัน

หน้าที่ของวงจร Control unit คือส่งสัญญาณควบคุมการทำงานของวงจรต่างๆ ภายใน Datapath ให้สามารถทำงานได้ตามที่ต้องการ วงจร Control Unit แบ่งออกได้เป็น 2 ประเภทคือแบบ Hardwired และแบบ Micro-programmed สไลด์หน้าที่ 55-57 จะแสดงตัวอย่างของสัญญาณควบคุมที่ต้องใช้ในการทำคำสั่ง 3 คำสั่งคือคำสั่ง ADD, Branch, Branch<0 โดยค่าสัญญาณควบคุมต่างๆ จะต้องนำไปคิดในขั้นตอนการออกแบบ Control Unit

สไลด์หน้าที่ 58 แสดงบล๊อกไดอะแกรมของคอนโทรลยูนิตแบบ Hardwired โดยจะมีวงจร Control Step Counter ทำหน้าที่เป็น Counter ซึ่งจะนับค่าขึ้นหนึ่งค่าเมื่อมีสัญญาณคล๊อกเข้ามา 1 คล๊อก วงจร Encoder จะรับค่าจาก Control Step Counter และคำสั่งจาก Instruction Register เข้ามาเพื่อเข้ารหัสเป็นสัญญาณควบคุม นอกจากนี้ในการทำงานของวงจร Control Unit ยังต้องรับค่าสัญญาณภายนอกและ Condition Code เข้ามาประมวลผลด้วย ตัวอย่างสัญญาณภายนอกเช่นสัญญาณ Memory Function Complete ในขั้นตอนการเฟตช์คำสั่ง และตัวอย่างสัญญาณจาก Condition code เช่นการตัดสินใจการบรานช์กรณีทำคำสั่งบรานช์แบบมีเงื่อนไข เป็นต้น

สไลด์หน้าที่ 59 ได้ขยายวงจรในหน้าที่ 58 ให้ละเอียดขึ้นโดยวงจร Control Step Counter จะแยกเป็นวงจร Counter และวงจร Step Decoder โดยหากเรามองย้อนไปยังสไลด์หน้า 55-57 จะเห็นว่าคำสั่งแต่ละคำสั่งจะประกอบไปด้วยหลายขั้นตอนในการทำงาน ซึ่งจะสังเกตเห็นว่าทั้งการทำงานของทั้งสามคำสั่งมีการส่งสัญญาณควบคุมในขั้นตอนที่ 1 ถึงขั้นตอนที่ 3 เหมือนกัน ดังนั้น วงจร Step decoder จะบอกวงจร Encoder ว่าในปัจจุบันซีพียูกำลังทำงานที่สเตทใด วงจร Encoder จะได้ส่งสัญญาณควบคุมออกไปได้ถูกต้องอย่างที่มันควรจะเป็น และแม้ว่าทุกคำสั่งจะต้องการสัญญาณควบคุมในส่วนของการเฟตช์เหมือนกันแต่การเอกซีคิวต์ของแต่ละคำสั่งจะใช้สัญญาณควบคุมต่างกันมาก ดังจะเห็นว่าในขั้นตอนที่ 4 ของการเอกซีคิวต์คำสั่ง ADD R1,[R3] นั้นจะต้องการสัญญาณที่แอก

ตีฟ 3 สัญญาณคือ R3out, MARin, Read ส่วนการเอกซีคิวต์คำสั่งบรานช์ในขั้นตอนที่ 4 กลับต้องการควบคุม 3 สัญญาณคือ Offset-field-of-IRout, ADD, Zin ดังนั้นหลังจากเฟตช์เสร็จแล้ววงจร Encoder จะต้องการรู้ด้วยว่าคำสั่งที่ต้องการเอกซีคิวต์เป็นคำสั่งใด ซึ่งวงจร Instruciton Decoder จะทำหน้าที่บอกวงจร Encoder ว่าในขณะนั้นๆ ซีพียูกำลังทำคำสั่งใด

ในการสร้างสัญญาณควบคุมจะต้องรู้สัญญาณที่แอกตีฟในแต่ละขั้นตอนของทุกคำสั่ง ตัวอย่างในสไลด์หน้าที่ 60 แสดงให้เห็นการสร้างสัญญาณควบคุม Zin ซึ่งจะเห็นได้ว่าสัญญาณ Zin จะแอกตีฟที่ T1 ของทุกคำสั่ง และแอกตีฟที่ T6 ของคำสั่ง ADD และแอกตีฟที่ T4 ของคำสั่ง Branch ดังนั้นวงจรในการสร้างสัญญาณ Zin ก็จะมีสมการเป็น Zin = T1 + (T6•ADD) + (T4•BR)+….. และสามารถสร้างเป็นวงจรดังในสไลด์หน้าที่ 61 ส่วนสัญญาณ END ก็ใช้วิธีการพิจารณาการแอกตีฟของแต่ละสเตทเช่นเดียวกัน วงจรของการสร้างสัญญาณ END แสดงในสไลด์หน้า 62

ในการสร้างคอนโทรลยูนิตแบบ Hardwired จะเห็นว่าเรานำวงจรฮาร์ดแวร์มาเชื่อมต่อ (Wiring) เข้าด้วยกัน ซึ่งวงจรที่ได้จะมีความซับซ้อนสูงมาก ดังนั้นในการสร้างวงจรควบคุมจึงมีผู้เสนอวิธีการอีกวิธีขึ้นมานั่นคือการใช้ micro program โดยในสไลด์หน้าที่ 63 แสดงบล๊อกไดอะแกรมของวงจรคอนโทรลยูนิตแบบนี้ หลักการทำงานคือ หากเราพิจารณาให้ดีจะเห็นว่าสัญญาณควบคุมของแต่ละสเตปการทำงานก็คือการส่งค่า 1 และค่า 0 ให้กับสัญญาณควบคุมนั้นๆ ดังแสดงในตารางของสไลด์หน้าที่ 64 ซึ่งจากตารางจะเห็นว่า ในสเตปที่ 1 มีสัญญาณที่เป็นลอจิก 1 อยู่ 6 เส้น และเส้นอื่นๆ เป็นลอจิกศูนย์ ดังนั้นจึงเกิดแนวคิดว่าแทนที่จะสร้างวงจรตรวจสอบการทำงานของแต่ละสเตทก็ให้สร้าง ROM ขึ้นมาเก็บค่าสถานะของสัญญาณควบคุมแต่ละสเตทไปเลย และให้ ROM 1 ตำแหน่งเก็บค่าสัญญาณควบคุมของ 1 สเตทของคำสั่งใดๆ ซึ่งเราเรียก 1 ตำแหน่งของข้อมูลใน ROM นี้ว่า Micro-instruction ในรูปของสไลด์หน้าที่ 63 จะเรียก ROM ว่าเป็น Control Store

การทำงานของ CU แบบนี้เริ่มจากจะต้องป้อนค่าแอดเดรสของ ROM ที่ต้องการอ่านสัญญาณควบคุมออกมาโดยวงจร Starting-and-branch-address generator เข้าที่ uPC (micro program counter) โดยค่าใน uPC จะเพิ่มขึ้นทีละ 1 ค่าเมื่อได้รับสัญญาณคล๊อกเข้ามา 1 ลูก ในบางครั้งค่าของ uPC อาจมีการบรานช์ได้ โดยวงจร Starting-and-branch-address generator อาจรับค่าของ คำสั่งเข้ามาเพื่อบรานช์ไปอ่าน micro-program ของคำสั่งที่ต้องการ
