บทที่ 4

การทำงานของวงจร

หัวข้อการทำงานของวงจรอินเซอร์กิตอีมูเลเตอร์ จากโครงสร้างทางฮาร์ดแวร์ที่แสดงในบทที่ 3 เราจะแยกอธิบายการทำงานของวงจรแต่ละส่วน ดังนี้ คือ

1. วงจรส่วนคอนโทรลซีพียู
2. วงจรติดต่อสื่อสาร RS-232C
3. การควบคุมการทำงานของซีพียู Z-80
4. การตรวจสอบการเริ่มต้นคำสั่งของไมโครโปรเซสเซอร์ Z-80
5. วงจรอีมูเลชันซีพียู Z-80
6. วงจรอีมูเลชันซีพียู 8085
7. วงจรหน่วยความจำอีมูเลชัน
8. วงจรควบคุมการหยุด
9. วงจรติดตามการทำงานในเวลาจริง

ในวงจรต่างๆนี้ จะใช้อุปกรณ์ PAL ในการออกแบบวงจร ในการอธิบายการทำงานจะอธิบายสัญลักษณ์ตรรก ซึ่งใช้ในโปรแกรม PALASM2 (Monolithic Memories) โดยมีรูปแบบทั่วไปดังนี้

สำหรับวงจรคอมบิเนชัน

OUTPUT = INPUT1+/INPUT2 * INPUT3 ; COMMENT

เมื่อ OUTPUT แทนชื่อขาสัญญาณออก

INPUT 1 INPUT 2 INPUT 3 แทนชื่อขาสัญญาณเข้าหรือสัญญาณป้อนกลับจากเอาต์พุต

*	แทนลอจิก AND
+	แทนลอจิก OR
/	แทนลอจิก NOT

ถ้าไม่มีวงเล็บกำกับ จะทำการ AND จบก่อนจึงทำการ OR

สำหรับขาสัญญาณออกที่เป็นแบบ 3 สถานะ จะมีสมการเพิ่มเติมดังนี้

OUTPUT.TRST = INPUT1 * INPUT2

เมื่อ INPUT1 * INPUT2 เป็นจริง OUTPUT จึงจะได้สัญญาณออกตามสมการแรก กรณีอื่น OUTPUT จะเป็นค่าอิมพีแดนซ์สูง

ถ้าไม่มีสมการนี้ OUTPUT จะมีค่าตามปรกติ

สำหรับวงจรซีเควนเชียล

OUTPUT := INPUT1 + INPUT2 * INPUT3

OUTPUT จะมีค่าตามสมการหลังจากขอบขาขึ้นของสัญญาณนาฬิกา

วงจรส่วนคอนโทรลซีพียู (รูป 4.1)

วงจรประกอบด้วยซีพียู 8085 มีวงจรรีเซตที่มีสวิทช์กดสำหรับให้ผู้ใช้ทำการเริ่มระบบใหม่เมื่อมีข้อผิดพลาดมาก วงจรกำเนิดสัญญาณนาฬิกา ใช้คริสตัลความถี่ 6.144 เมกกะเฮิรตซ์ ที่จะให้ความถี่ที่ขาสัญญาณนาฬิกาขาออก 3.075 เมกกะเฮิรตซ์ สำหรับผลิตความถี่และสำหรับกำหนดความเร็วในการรับส่งแบบอนุกรมได้พอดี ชื่อสัญญาณทั้งหมดของซีพียู นี้จะตั้งเป็นชื่อนำหน้าด้วย C_ เพื่อบอกว่าเป็นสัญญาณของคอนโทรลซีพียู

อุปกรณ์ประกอบของซีพียูที่จำเป็นคือ รอม 27256 ซึ่งต่อแอดเดรสไบต์ต่ำกับสัญญาณแอดเดรสที่แลตช์จากสัญญาณแอดเดรสข้อมูลของ 8085 อุปกรณ์ที่ทำหน้าที่นี้คือ 74HCT573 สำหรับแรม ใช้ 8155 ซึ่งออกแบบสำหรับ 8085 โดยเฉพาะจึงไม่ต้องมีวงจรแลตช์และขาต่างๆ จะต่อกับสัญญาณจากซีพียูได้โดยตรง

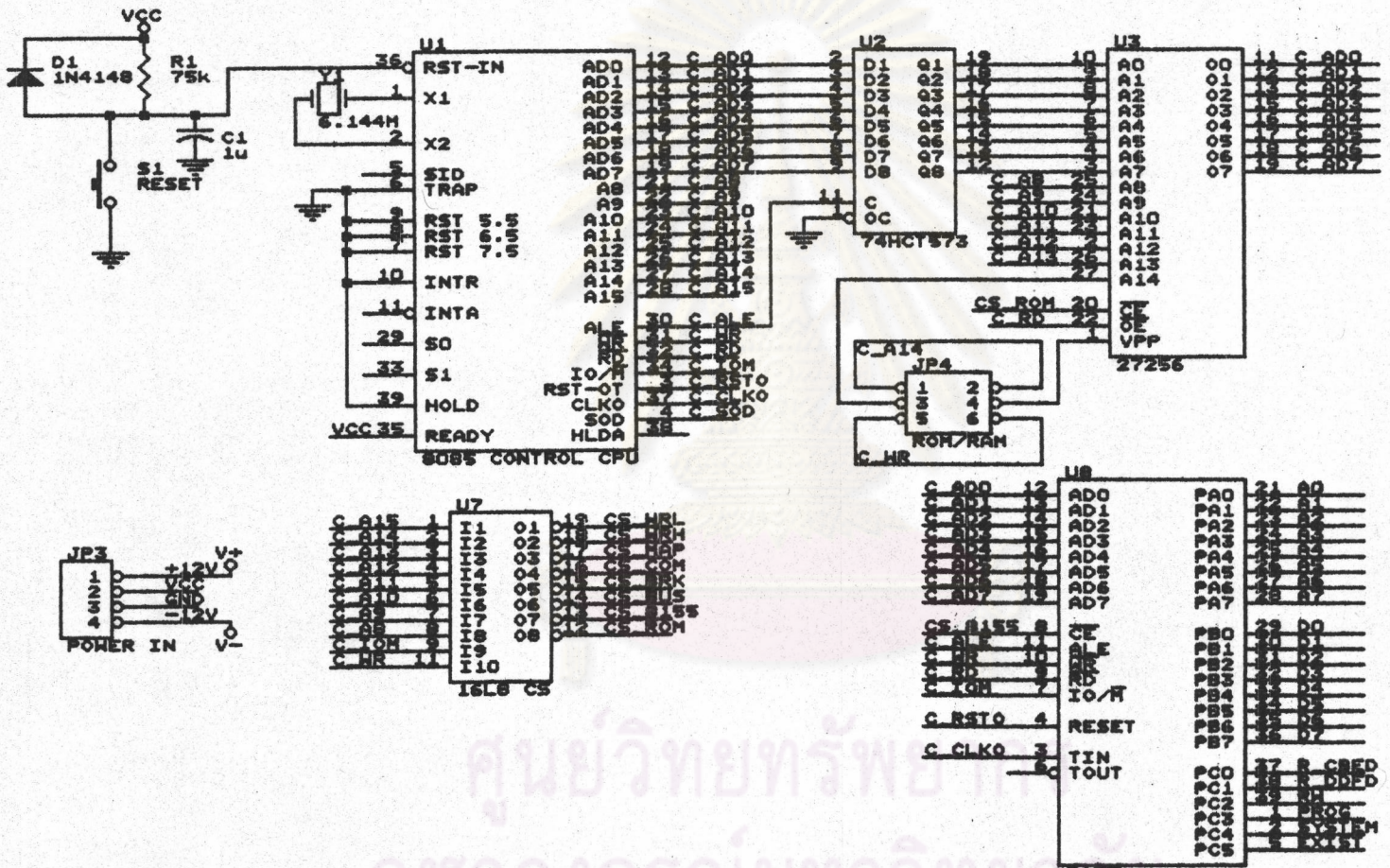
เนื่องจากอินเซอร์กิตอีมูเลเตอร์สำหรับ 8085 ที่ใช้เป็นเครื่องมือพัฒนาวงจรนี้มีหน่วยความจำอีมูเลชันเนียง 8 กิโลไบต์ ซึ่งไม่พอสำหรับการทดสอบซอฟต์แวร์ จึงใช้แรมต่อแทนรอมบนวงจรในระหว่างพัฒนา โดยใช้แรม 62256 ขนาด 32 กิโลไบต์ ตำแหน่งขาของแรมและรอมมีที่ต่างกัน 2 แห่ง จึงต้องมีขั้วต่อเพื่อสลับสัญญาณที่จะไปเข้าขา 1 และขา 27 ของหน่วยความจำรอม/แรมตัวนี้ ถ้าต้องการใช้รอมจะต่อขา 1 กับสัญญาณ C_WR คือต่อขา 4 กับขา 6 ของขั้วต่อที่ชื่อ ROM/RAM และต่อขา 27 กับสัญญาณ C_A14 คือต่อขา 1 กับขา 3 ของขั้วต่อ

เมื่อใช้รอม ขา 1 ต่อกับ C_WR ขา 27 ต่อกับ C_A14

ดังนั้นขั้วต่อที่ชื่อ ROM/RAM จะต่อขา 1 กับขา 3 และขา 4 กับขา 6

เมื่อใช้แรมขา 1 ต่อกับ C_A14 ขา 27 ต่อกับ C_WR

ดังนั้นขั้วต่อที่ชื่อ ROM/RAM จะต่อขา 3 กับขา 5 และขา 2 กับขา 4



รูป 4.1 วงจรคอนโทรลซีพียู

วงจรถอดรหัสแอดเดรส สำหรับเลือกอุปกรณ์ต่างๆ ใช้ไอซี PAL16L8 ซึ่งตั้งชื่อว่า CS ได้สัญญาณออกสำหรับอุปกรณ์ต่างๆ ดังนี้

CS_ROM เลือกใช้หน่วยความจำรอม/แรม (27256/62256)

CS_8155 เลือกใช้หน่วยความจำ/อินพุตเอาต์พุตของ 8155

CS_BUS เลือกใช้อินพุตเอาต์พุตสำหรับ 8255 ที่ตั้งชื่อว่า BUS

CS_BRK เลือกใช้อินพุตเอาต์พุตสำหรับ 8255 ที่ตั้งชื่อว่า BRK

CS_COM เลือกใช้อินพุตเอาต์พุตสำหรับ 8251 ที่ตั้งชื่อว่า COM

CS_MAP เลือกใช้เอาต์พุตสำหรับเขียนหน่วยความจำ 74LS189 ที่ใช้ในวงจรเลือกใช้และกำหนดลักษณะของหน่วยความจำ

CS_WRH เลือกใช้อินพุตเอาต์พุตสำหรับวงจรติดตามการทำงานของซีพียู

CS_WRL เลือกใช้อินพุตเอาต์พุตสำหรับวงจรติดตามการทำงานของซีพียู

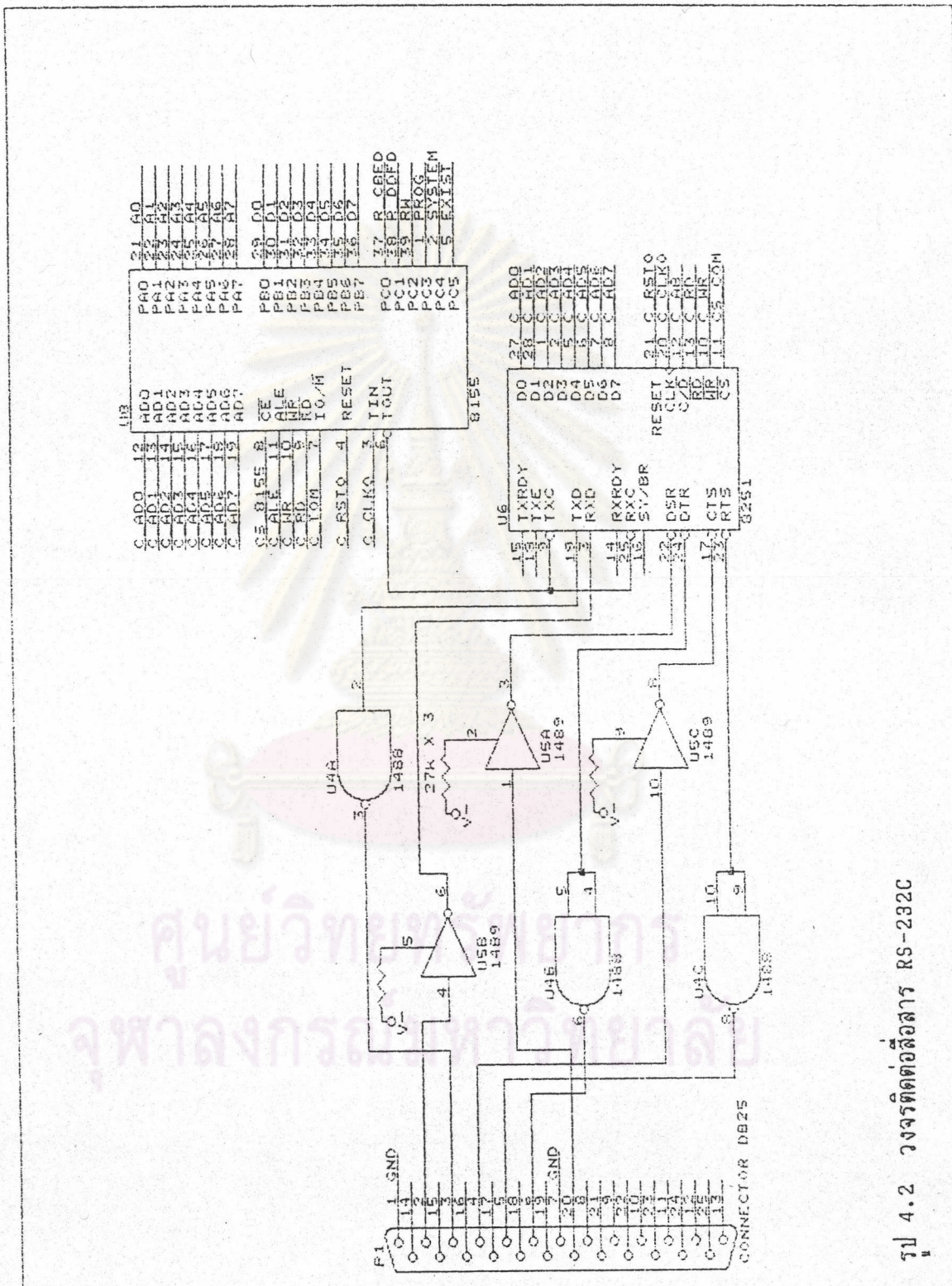
แต่ละอุปกรณ์มีตำแหน่งแอดเดรสและสถานะดังนี้

CS_ROM	0000 - 7FFF	MEMORY
CS_8155	8000 - 87FF	MEMORY/IO
CS_BUS	88 - 8B	IO
CS_BRK	8C - 8F	IO
CS_COM	90 - 91	IO
CS_MAP	86	IO WRITE
CS_WRH	94 - 97	IO
CS-WRL	98 - 98	IO

<u>วงจรติดต่อสื่อสาร RS-232C</u> (รูป 4.2)

ประกอบด้วย 8155 ที่กล่าวถึงแล้วในวงจรส่วนคอนโทรลซีพียู ในที่นี้จะให้สังเกตเฉพาะขา TIN ซึ่งต่อกับสัญญาณ C_CLKO ของซีพียูกับขา TOUT ซึ่งเป็นขาสัญญาณออกจากวงจรนับภายใน 8155 เราสามารถโปรแกรมให้สัญญาณนี้เป็นค่าความถี่ของสัญญาณนาฬิกาที่ออกจากซีพียูหารด้วยจำนวนเต็ม

วงจรที่ทำหน้าที่ติดต่อสื่อสารโดยรับส่งข้อมูลแบบอนุกรม คือ 8251 ซึ่งต่อกับสัญญาณจากซีพียูตามที่แสดงในรูปโดยขา TXC และ RXC ที่กำหนดอัตราการส่งรับข้อมูลแบบอนุกรมได้มา



รูป 4.2 วงจรติดต่ออุปกรณ์สื่อสาร RS-232C

จากสัญญาณออกจากวงจรนับ 8155 ในงานนี้เราต้องการใช้อัตราการรับส่งข้อมูล 9600 บิตต่อวินาที และโปรแกรมให้ 8251 ใช้สัญญาณกำหนดอัตราส่งรับข้อมูลภายนอกเป็น 16 เท่าของอัตราการรับส่งข้อมูลจริง เราจึงต้องกำหนดตัวหารความถี่ในวงจรนับของ 8155 ดังนี้

ความถี่ของสัญญาณเข้า 3.072 เมกกะเฮิรตซ์

ความถี่ของสัญญาณออก 9600 x 16 เฮิรตซ์

ตัวหารเท่ากับ = 3072000/(9600 x 16) = 20

วงจรเปลี่ยนระดับสัญญาณทีทีแอลที่ 8251 รับรู้ให้เข้ากับสัญญาณตามมาตรฐาน RS-232C คือ ไอซี 1488 และ 1489 ซึ่งหาซื้อได้ง่ายกว่าไอซีอื่นๆ ที่ทำหน้าที่นี้

ขั้วต่อสาย RS-232C ที่ใช้ในวงจรนี้จะต่อเหมือนกับวงจรโมเดม คือ ขา 2 เป็นสัญญาณเข้า ขา 3 เป็นสัญญาณออก ขั้วต่อที่อยู่บนเครื่องเป็นแบบ DB25 ตัวเมีย การกำหนดขั้วต่อลักษณะนี้ทำให้ต่อกับเครื่องคอมพิวเตอร์ได้โดยใช้สายต่อแบบต่อตรง ซึ่งหาซื้อได้ทั่วไป ป้องกันปัญหาในการใช้สายผิด แล้วทำให้อุปกรณ์ในวงจรเสียหายได้

วงจรที่อธิบายมาทั้งสองส่วนสามารถใช้งานร่วมกันได้ทั้งอินเซอร์กิตอีมูเลเตอร์ สำหรับ Z-80 และ 8085 แต่วงจรในส่วนต่อจากนี้จะมีความแตกต่างกันในการใช้กับซีพียูแต่ละเบอร์ การอธิบายในวงจรที่มีความแตกต่างกันไม่มากจะแสดงในรูปเดียวกัน แต่สร้างกรอบจุดประให้เห็นส่วนที่เป็นวงจรเฉพาะของ Z-80 และ 8085 วงจรที่มีความแตกต่างกันมากจะแสดงรูป และคำอธิบายแยกจากกัน

<u>การควบคุมการทำงานของซีพียู Z-80</u>

จากที่กล่าวในเรื่องทฤษฎีและหลักการที่เลือกใช้แล้วว่าเราจะสร้างสัญญาณ WAIT ให้กับซีพียู ในการทำงานทีละคำสั่งและการทำให้ซีพียูหยุดการทำงาน นอกจากนี้เรายังใช้ช่วงเวลาที่ซีพียูหยุดรอนี้เปลี่ยนการติดต่อระหว่างซีพียูกับหน่วยความจำที่ผู้ใช้กำหนดเป็นการติดต่อกับหน่วยความจำที่ใช้อีมูเลชันซีพียู อ่านคำสั่งแต่ช่วงเวลาที่ซีพียูหยุดรอนี้ สามารถยืดได้นานเท่าที่ต้องการจึงไม่จำเป็นต้องใช้ หน่วยความจำจริงๆ ในที่นี้เราต้องมีอินพุตพอร์ตของคอนโทรลซีพียูที่ใช้อ่านสัญญาณจากบัสข้อมูล ถ้าอินพุตนี้สามารถโปรแกรมให้เปลี่ยนเป็นเอาต์พุตได้หลังจากตัดการติดต่อหน่วยความจำที่ต่อกับบัสข้อมูลแล้ว เราก็ใช้ค่าที่ส่งออกเอาต์พุตเป็นโปรแกรมควบคุมอีมูเลชันซีพียูได้เช่นกัน

โปรแกรมที่เราส่งจากเอาต์พุตพอร์ตไปยังบัสข้อมูลเพื่อควบคุมอีมูเลชันซีพียูที่สำคัญ คือ

โปรแกรมคืนค่ารีจิสเตอร์และโปรแกรมเก็บค่ารีจิสเตอร์ที่จะต้องทำก่อนและหลังโมดอีมูเลชัน ตัวอย่างของโปรแกรมเก็บค่ารีจิสเตอร์ที่นิยมใช้ คือคำสั่ง PUSH AF ซึ่งจะนำข้อมูลในรีจิสเตอร์ AF มาเขียนใส่สแตก (แต่จะไม่ถูกเขียนที่หน่วยความจำเพราะยกเลิกการติดต่อแล้ว) ขั้นตอนการทำงานของ Z-80 กับคำสั่งนี้เป็นดังนี้ (รูป 4.3)

ซีพียูอยู่ในช่วงที่กำลังจะอ่านรหัสคำสั่งใหม่ แต่ได้รับสัญญาณ WAIT

อินพุตพอร์ตที่ต่อกับบัสข้อมูลถูกโปรแกรมเป็นเอาต์พุตพอร์ต

เอาต์พุตพอร์ตส่งรหัส 0F5H ไปที่บัสข้อมูล

วงจรควบคุมยกเลิกสัญญาณ WAIT ซีพียูทำงานต่อได้

ซีพียู อ่านรหัส 0F5H แปลว่า PUSH AF

ในแมชชีนไซเคิลต่อไป ซีพียูจะส่งข้อมูลจากรีจิสเตอร์ A มาที่บัสข้อมูล ในช่วงนี้จะพบว่าสัญญาณบนบัสข้อมูลเกิดความไม่แน่นอน เนื่องจากเอาต์พุตพอร์ตยังคงส่งสัญญาณ 0F5H ออกมาและบัสข้อมูลของซีพียูก็มีสถานะเป็นเอาต์พุตเช่นกัน

จากการพิจารณาในจุดนี้ ทำให้วงจรง่ายๆ ที่เราออกแบบไม่สามารถใช้งานได้จริง ถ้าในกรณีที่อุปกรณ์ที่ให้สัญญาณ 0F5H เป็นหน่วยความจำจริงๆ จะไม่เกิดปัญหานี้ เพราะว่าสัญญาณออกจากหน่วยความจำจะถูกควบคุมให้เป็นสัญญาณออกได้ในเวลาที่ซีพียูส่งสัญญาณขออ่านหน่วยความจำเท่านั้น

พิจารณาจากรูป 4.3 เราอาจใช้วิธีแก้ปัญหาแบบที่ใช้กับหน่วยความจำ คือ ให้สัญญาณจากเอาต์พุตพอร์ต ถูกควบคุมด้วยสัญญาณอ่านจากซีพียู วิธีนี้จะทำให้เราไม่อาจใช้พอร์ตมาตรฐานที่สามารถโปรแกรมให้เปลี่ยนอินพุตเป็นเอาต์พุตมาทำหน้าที่นี้ให้ ต้องใช้พอร์ตเอาต์พุตโดยเฉพาะที่มีสัญญาณควบคุมเอาต์พุตให้เป็นแบบอิมพีแดนซ์สูง วงจรจึงมีความยุ่งยากเพิ่มมากขึ้น

อีกวิธีหนึ่งคือ โปรแกรมพอร์ตเอาต์พุตเป็นอินพุตในช่วงที่ซีพียูอ่านสัญญาณเสร็จก่อนที่ซีพียูจะเริ่มแมชชีนไซเคิลใหม่ ซึ่งเราทำได้โดยการหน่วงเวลาระหว่างแมชชีนไซเคิลเหมือนกับที่ใช้สัญญาณ WAIT ทำให้เกิดการหน่วงเวลาในแมชชีนไซเคิล

ดังนั้น ในการควบคุมการทำงานของไมโครโปรเซสเซอร์ในที่นี้เราจะต้องใช้สัญญาณ BUSRQ ร่วมด้วย

ขั้นต่อไปเราจะพิจารณาว่าสัญญาณ WAIT และ BUSRQ มีจังหวะการทำงานอย่างไรจึงจะควบคุมไมโครโปรเซสเซอร์ได้ตามต้องการเพื่อใช้ออกแบบวงจรส่วนนี้

เมื่อซีพียูทำงานทีละคำสั่ง เราใช้ประโยชน์จากสัญญาณ WAIT โดยให้ซีพียูเริ่มทำงานในการอ่านคำสั่งจากหน่วยความจำที่ผู้ใช้กำหนดขณะสัญญาณ WAIT เป็น 0 เมื่อผู้ใช้สั่งให้ทำงาน

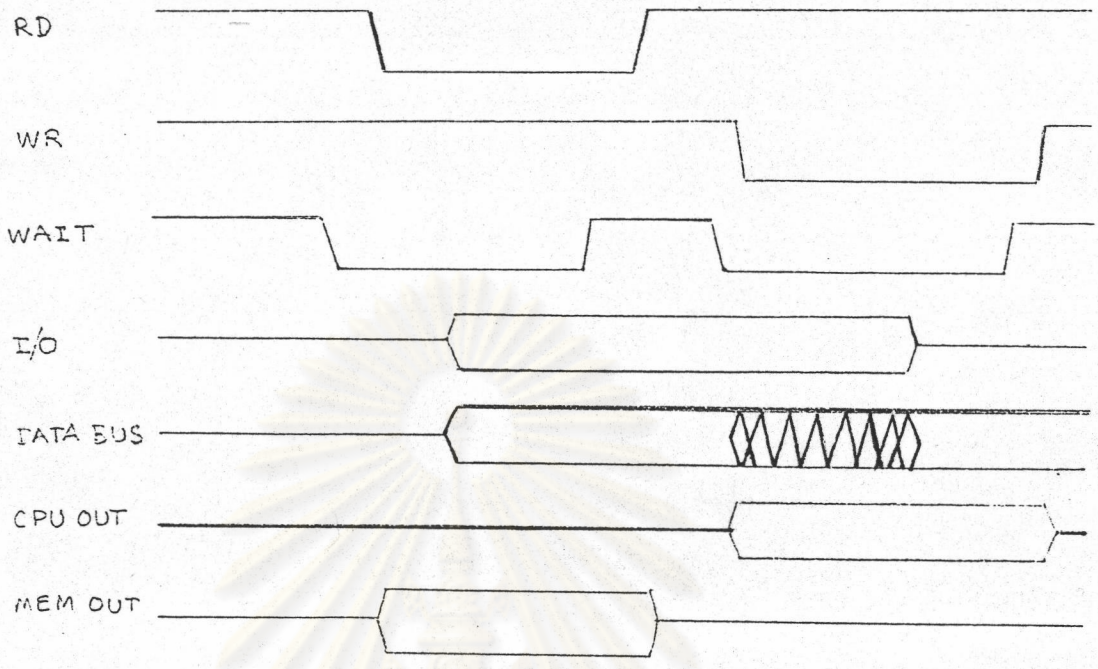
ในคำสั่งต่อไป วงจรควบคุมจะทำให้ WAIT กลายเป็น 1 ซีพียูจะอ่านคำสั่งโปรแกรมแล้วทำงานต่อไปในแมชชีนไซเคิลต่อไป วงจรควบคุมจะต้องสร้างสัญญาณ WAIT ให้เป็น 0 อีกครั้งเพื่อให้ซีพียูหยุดรอในแมชชีนไซเคิลต่อไป สัญญาณ WAIT จะต้องเป็น 0 ก่อนที่ซีพียูจะทำงานถึงช่วงสัญญาณนาฬิกาที่ตรวจสอบสัญญาณนี้ วิธีที่ง่ายที่สุด คือ หลังจากทำให้ WAIT เป็น 1 ด้วยการควบคุมจากผู้ใช้แล้ว WAIT จะถูกทำให้เป็น 0 อีกครั้งเมื่อการอ่านเขียนหน่วยความจำในแมชชีนไซเคิลนั้นจบลง โดยสัญญาณควบคุมจากผู้ใช้จะไม่มีผลในช่วงนี้

ในการทำเป็นวงจร เราจะใช้ฟลิปฟลอปทำหน้าที่รับสัญญาณควบคุมปรกติ เมื่อไม่ได้ทำงานด้วยความเร็วเต็มที่ ซีพียูจะถูกทำให้ได้รับสัญญาณ WAIT เป็น 0 เราจะให้เอาต์พุตของฟลิปฟลอปเป็น 0 เมื่อผู้ใช้สั่งให้ทำงานในแมชชีนไซเคิลนี้ให้เสร็จแล้วข้ามไปทำงานในแมชชีนไซเคิลถัดไป ฟลิปฟลอปจะถูกทำให้เป็น 1 โดยสัญญาณขอบขาขึ้นของสัญญาณควบคุมที่ทำงานตามผู้ใช้สั่ง เมื่อจบการอ่านหรือเขียนซีพียูจะให้สัญญาณอ่านเขียนเปลี่ยนเป็น 1 ทั้งคู่ สัญญาณนี้จะทำให้ฟลิปฟลอปกลายเป็น 0 อีกครั้ง ซึ่งซีพียูจะหยุดรอในแมชชีนไซเคิลถัดไปตามต้องการ สัญญาณที่ผู้ใช้สั่งจะต้องกลายเป็น 0 แล้วเปลี่ยนเป็น 1 อีกครั้งจึงจะทำงานในแมชชีนไซเคิลนั้น ลักษณะสัญญาณและวงจรแสดงในรูป 4.4 สัญญาณที่ผู้ใช้สั่ง ในที่นี้สร้างจากเอาต์พุตพอร์ตของคอนโทรลซีพียู

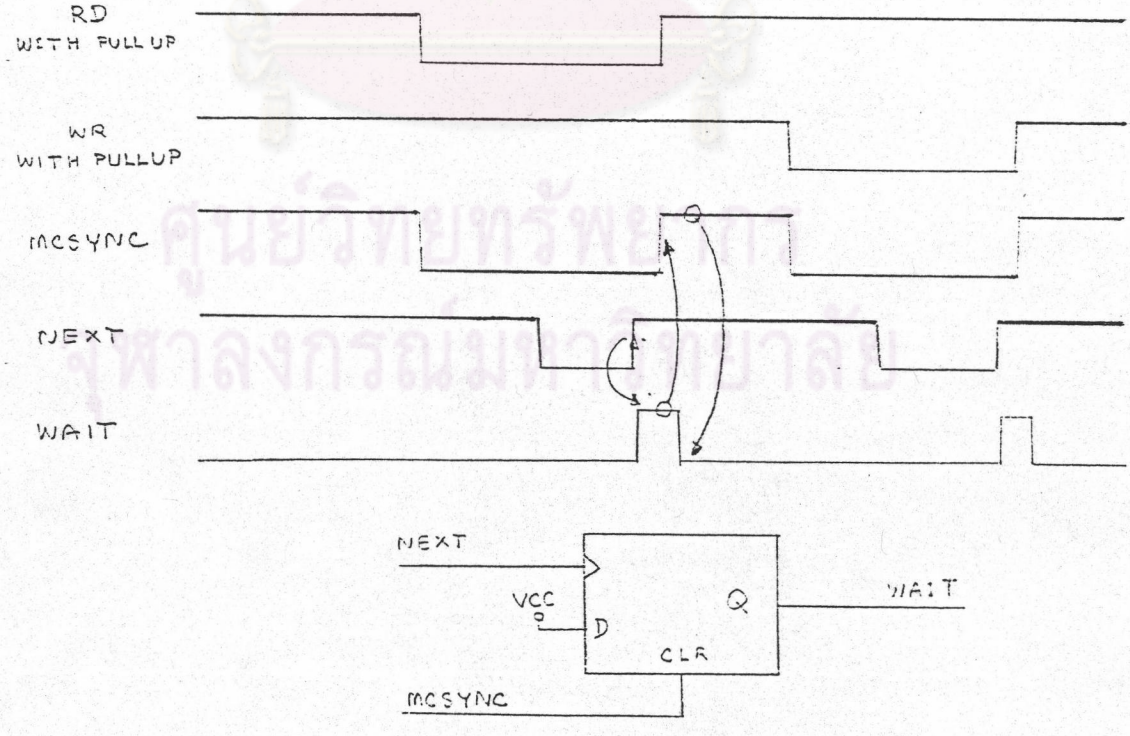
จากตัวอย่างเรื่องการใช้ซีพียูรับคำสั่ง PUSH AF ในโมดอินเทอร์โรเกชัน เราต้องการสัญญาณ BUSRQ เพิ่มเข้ามาหลังจากซีพียูอ่านรหัสคำสั่งแล้ว สัญญาณนี้สามารถควบคุมจากการสั่งของผู้ใช้ได้โดยตรง (จากพอร์ตเอาต์พุตของคอนโทรลซีพียู) แสดงรูปสัญญาณในรูป 4.5

ถ้าเราใช้การทำงานของสัญญาณ WAIT และ BUSRQ ในลักษณะนี้กับการทำให้อีมูเลชันซีพียูอ่านเขียนข้อมูลหน่วยความจำของผู้ใช้ในโมดอินเทอร์โรเกชันจะมีขั้นตอนแสดงในรูป 4.6 ซึ่งแสดงให้เห็นว่าเราต้องอ่านเขียนหน่วยความจำในโมดอินเทอร์โรเกชัน (จากคำสั่งของผู้ใช้ที่แสดงและแก้ไขหน่วยความจำ) ด้วยการมีสัญญาณ WAIT ในแมชชีนไซเคิลที่ติดต่อหน่วยความจำ ผลการอ่านเขียนจึงอาจผิดพลาดจากที่ควรจะเป็นเวลาใช้งานจริงได้ เช่นการใช้หน่วยความจำความเร็วต่ำ เมื่อทดลองอ่านเขียนด้วยคำสั่งของอินเซอร์กิตอีมูเลเตอร์ แล้วใช้งานได้เพราะมีสถานะ WAIT เพิ่มขึ้นมาก แต่พอทำงานด้วยเวลาจริงตามโปรแกรมผู้ใช้ แล้วอาจจะอ่านเขียนไม่ได้

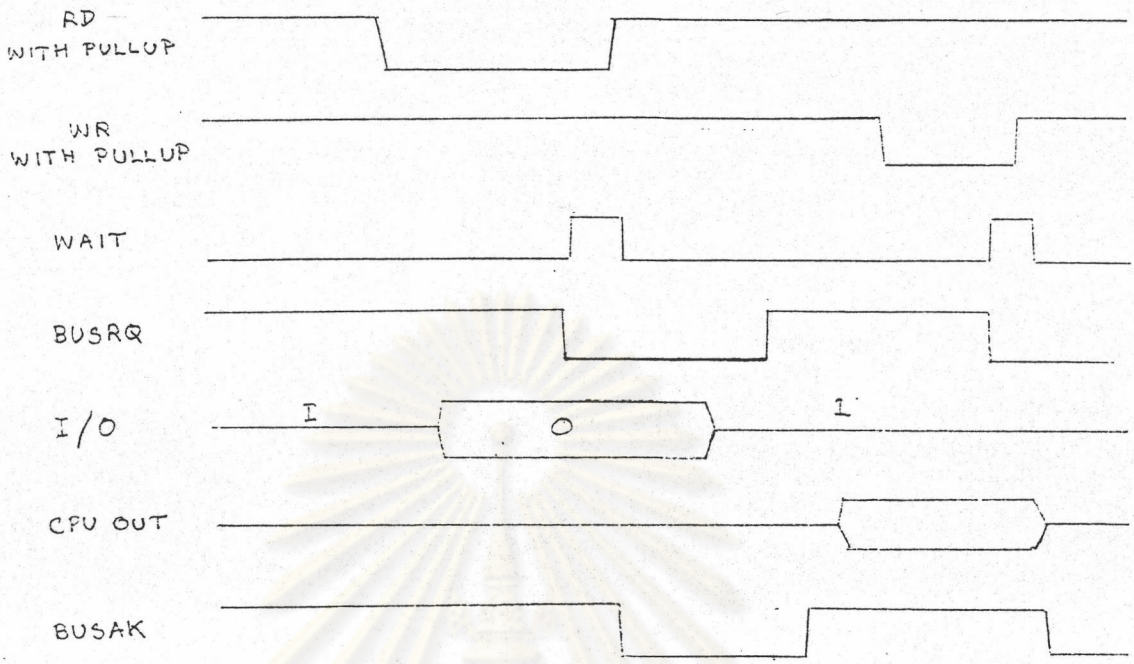
เราจะเพิ่มความถูกต้องของการทำงานส่วนนี้ได้โดยให้วงจรสร้างสัญญาณ BUSRQ มีลักษณะเดียวกับสัญญาณ WAIT แต่ให้สัญญาณ BUSRQ ถูกทำให้เป็น 0 ด้วยการที่มีสัญญาณอ่านหรือเขียนเป็น 0 แสดงลักษณะวงจรและรูปสัญญาณในรูป 4.7



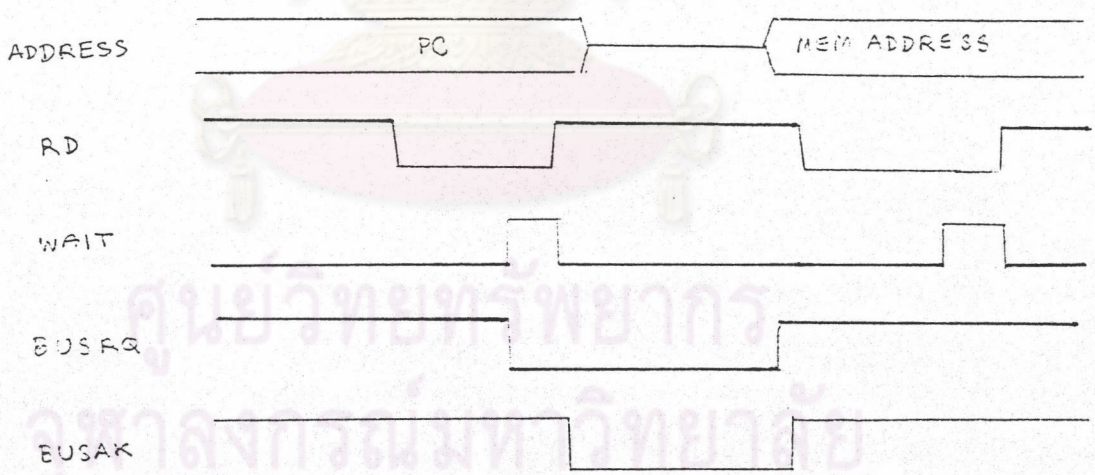
รูป 4.3 ปัญหาจากการใช้สัญญาณ WAIT เพียงอย่างเดียว



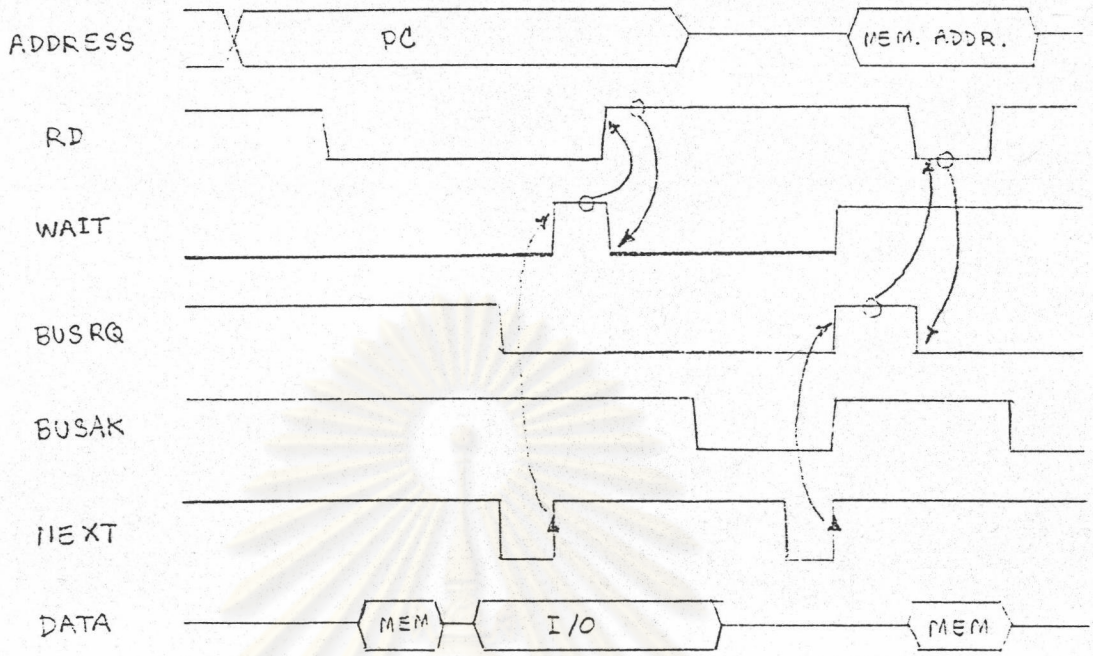
รูป 4.4 วงจรฟลิปฟลอปสร้างสัญญาณ WAIT

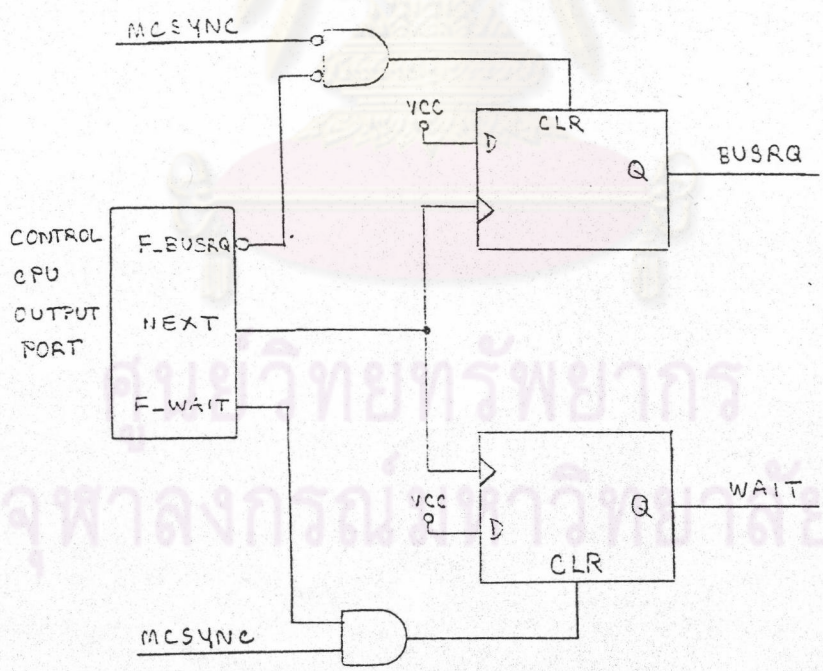


รูป 4.5 การใช้สัญญาณ WAIT ร่วมกับ BUSRQ



รูป 4.6 ปัญหาการใช้สัญญาณ WAIT ร่วมกับ BUSRQ





/MCSYNC = /RD + /WR + /INTA

รูป 4.7 วงจรฟลิปฟลอปสร้างสัญญาณ BUSRQ

สำหรับซีพียู 8085 มีรูปแบบการอ่านเขียนบัสแบบเดียวกับ Z-80 เพียงแต่เรียกชื่อสัญญาณ WAIT ของ Z-80 ด้วยชื่อสัญญาณ READY และสัญญาณ BUSRQ ของ Z-80 กลายเป็น HOLD ใน 8085 และทำให้บัสกลายเป็นสถานะอิมพีแดนซ์สูง เมื่อสัญญาณ HOLD เป็น 1 ซึ่งตรงข้ามกับ Z-80 การสร้างสัญญาณ READY และ HOLD ของ 8085 จึงใช้หลักการเดียวกับที่กล่าวมาแล้ว

<u>การตรวจสอบการเริ่มต้นคำสั่งของไมโครโปรเซสเซอร์ Z-80</u>

จากการควบคุมไมโครโปรเซสเซอร์ด้วยสัญญาณ WAIT ที่ได้อธิบายมาแล้ว เมื่อนำมาใช้งานจริงในการหยุดไมโครโปรเซสเซอรีที่ทำงานตามโปรแกรมของผู้ใช้ตามเวลาจริง เมื่อพบจุดหยุดวงจรควบคุมสามารถทำให้ซีพียูหยุดได้ทันที แต่เราจะเริ่มเข้าสู่โมดอินเทอร์โรเกชันได้เมื่อซีพียูหยุดรอในแมชชีนไซเคิลที่จะรับคำสั่งใหม่เข้ามา เพราะในโมดอินเทอร์โรเกชันจะเริ่มต้นด้วยการให้ซีพียูอ่านคำสั่งเก็บค่ารีจิสเตอร์

สำหรับ 8085 สัญญาณที่แสดงสถานะของซีพียูว่าอยู่ในช่วงรับคำสั่ง คือ S0 = 1, S1 = 1, IO/M = 0 ซึ่งขณะที่อิมูเลชันซีพียูอยู่ในช่วง WAIT สามารถตรวจสอบได้ง่ายจากอินพุตพอร์ตของคอนโทรลซีพียู ถ้าพบว่าอิมูเลชันซีพียูยังไม่อยู่ในแมชชีนไซเคิลที่พร้อมจะรับคำสั่งก็จะให้ทำงานต่อไปอีก 1 แมชชีนไซเคิล จนกว่าจะพบแมชชีนไซเคิลที่พร้อมจะรับคำสั่งจึงเข้าสู่โมดอินเทอร์โรเกชัน

สำหรับ Z-80 สัญญาณที่แสดงสถานะของซีพียูว่าอยู่ในช่วงรับคำสั่ง คือ M1 = 0 และ RD = 0 แต่ Z-80 มีคำสั่งที่มีออพโค้ด 2 ไบต์ ขณะซีพียูอ่านออพโค้ดทั้ง 2 ไบต์ จะให้สัญญาณ M1 เป็น 0 ทั้ง 2 แมชชีนไซเคิล ถ้าเราใช้ M1 ในการตรวจสอบอาจจะทำให้เราเข้าโมดอินเทอร์โรเกชัน โดยซีพียูกำลังรออ่านออพโค้ดที่ 2 ของคำสั่ง คำสั่งที่เราให้ซีพียูอ่านในโมดนี้จึงกลายเป็นออพโค้ดที่ 2 ต่อจากที่ผู้ใช้อ่านค้างไว้แล้วแปลความหมายเป็นคำสั่งอื่นไป จึงทำงานผิดพลาด

จากการศึกษาเรื่องคำสั่งของ Z-80 ที่มีออพโค้ด 2 ไบต์ โดยดูจากชุดคำสั่งทั้งหมดพบว่าคำสั่งที่มีออพโค้ด 2 ไบต์ จะมีออพโค้ดไบต์แรกเป็น 0CBH, 0EDH, 0DDH หรือ 0FDH จึงนำอินเซอร์กิตอิมูเลเตอร์ MICE I สำหรับ Z-80 (Microtek International Inc., 1984) มาทดลองติดตามการทำงานของคำสั่งเหล่านี้โดยให้ทำงานทีละแมชชีนไซเคิล (MICE I ใช้หลักการควบคุมการทำงานด้วยสัญญาณ WAIT จึงทำงานทีละแมชชีนไซเคิลได้)

การทดลองใช้วิธีเตรียมออพโค้ดไบต์แรกไว้ ตามด้วยไบต์ 2 แบบต่างๆแล้วส่งสัญญาณ

NMI ในช่วงที่อ่านออพโค้ดแรก ทำให้เกิดการอินเตอร์รัพต์ที่ Z-80 ขึ้นและรอจนจบคำสั่งแล้วจึงตอบรับอินเตอร์รัพต์ โดยเขียนแอดเดรสของโปรแกรมที่จะทำงานต่อไว้ในสแตกแล้วไปทำงานที่แอดเดรส 0066H เราดูผลว่าคำสั่งออพโค้ด 2 ไบต์ จะทำงานจบคำสั่งเมื่อใด พบว่าถ้าออพโค้ดแรกเป็น OCBH หรือ OEDH แล้วออพโค้ดที่ตามมารวมกันได้เป็นคำสั่ง 1 คำสั่ง แมชชีนไซเคิลต่อมาที่มีสัญญาณ M1 จะเป็นการเริ่มคำสั่งใหม่ทันที ถ้าออพโค้ดแรกเป็น ODDH หรือ OFDH แล้วออพโค้ดที่ตามมารวมกันไม่ได้ คำสั่งที่ถูกต้อง Z-80 จะถือว่าออพโค้ดที่ 2 นี้เป็นการเริ่มต้นคำสั่งใหม่ ดังนั้นถ้าออพโค้ดที่ 2 นี้เป็น ODDH หรือ OFDH Z-80 จะไม่สามารถทำงานให้จบคำสั่งได้ใน 2ไบต์ แมชชีนไซเคิลต่อไปจะมี M1 อีกและ Z-80 รอรับคำสั่งออพโค้ดที่ 2 ต่อไป

จากการทดลองเราจะต้องหาวงจรมาทำหน้าที่ตรวจสอบการอ่านออพโค้ดก่อนหน้านั้นว่าเป็นออพโค้ดแรกของคำสั่งที่มีออพโค้ด 2 ไบต์หรือไม่ ถ้าเราหยุดโปรแกรมในโมดอีมูเลชันแล้วอ่านพบสถานะนี้จากวงจร เรายังไม่สามารถเปลี่ยนเป็นโมดอินเทอร์โรเกชันได้ต้องปล่อยให้ซีพียูทำงานทีละแมชชีนไซเคิลต่อไปจนพบว่าซีพียูกำลังอ่านออพโค้ดเริ่มต้นคำสั่งจริงๆจึงเปลี่ยนโมดได้

การทำงานของวงจรประกอบด้วย ฟลิปฟลอป 2 ตัว เพื่อจำสถานะของออพโค้ด OCBH หรือ OEDH กับอีกตัวสำหรับ ODDH หรือ OFDH

เนื่องจากถ้าออพโค้ดแรกเป็น OCBH หรือ OEDH ไม่ว่าออพโค้ดต่อมาเป็นอะไรคำสั่งก็จะจบภายใน 2 ออพโค้ดนี้ เราจึงใช้ค่าที่ฟลิปฟลอปจำค่านี้ไว้ป้อนกลับมาทำให้ฟลิปฟลอปในสถานะถัดไปไม่แสดงผลว่าเป็นออพโค้ดที่ 2

สัญญาณนาฬิกาที่ทำให้เกิดการเปลี่ยนสถานะของฟลิปฟลอปได้มาจากสัญญาณที่ใช้อ่านเขียนบัสของซีพียู คือ สัญญาณอ่าน, เขียน หรือตอบรับอินเตอร์รัพต์

วงจรทั้งหมดแสดงในรูป 4.8 โดยมีอุปกรณ์ที่สำคัญ คือ PAL16L8 ที่ชื่อ SYNCZ80

D0 - D7 คือ ข้อมูลของซีพียู Z-80

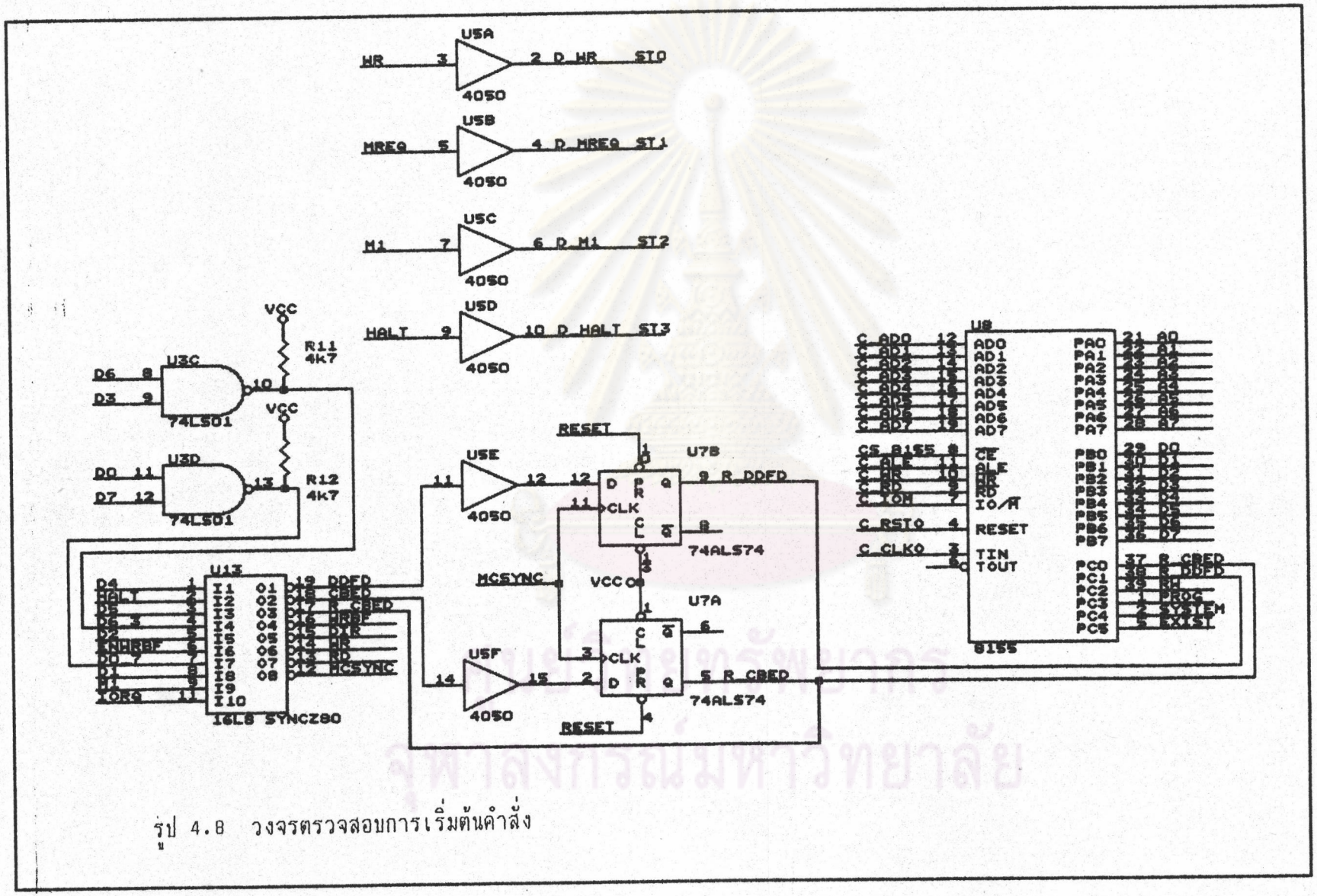
RD, WR, M1, IORQ คือ สัญญาณควบคุมจาก Z-80 ทำงานที่ลอจิก 0

RESET คือ สัญญาณจากวงจรควบคุมสำหรับรีเซต Z-80 เพื่อรีเซตฟลิปฟลอป

DDFD สัญญาณที่แสดงว่ามีการอ่านออพโค้ด ODDH หรือ OFDH และออพโค้ดก่อนหน้าไม่ใช่ OCBH หรือ OFDH

CBED สัญญาณที่แสดงว่ามีการอ่านออพโค้ด OCBH หรือ OFDH และออพโค้ดก่อนหน้าไม่ใช่ OCBH หรือ OFDH

ไอซี 4050 ที่ต่อระหว่าง DDFD และ CBED กับฟลิปฟลอปใช้หน่วงเวลาสัญญาณเพื่อให้คงอยู่ถึงช่วงที่มีสัญญาณอ่านเขียนบัสถึงขอบขาขึ้น



รูป 4.8 วงจรตรวจสอบการเริ่มต้นคำสั่ง

MCSYNC สัญญาณที่แสดงว่าซีพียูมีการอ่านหรือเขียนบัส ทำงานที่ลอจิก 0

R_DDFD และ R_CBED ใช้บอกคอนโทรลซีพียูว่าแมชชีนไซเคิลก่อนหน้านี้เป็นออพโค้ดของคำสั่งที่มีออพโค้ด 2 ไบต์ ต่อกับพอร์ตอินพุต 8155

อีมูเลชันซีพียูจะเปลี่ยนกลับไปโมดอินเทอร์โรเกชันได้เมื่อทั้ง R_DDFD และ R_CBED เป็น 1

สมการ PAL SYNCZ80 ที่เกี่ยวข้องกับหน้าที่ของวงจรที่อธิบายในส่วนนี้ ได้แก่

```
;       CB = 1100 1011
;            D7*  D6* /D5* /D4*  D3* /D2*  D1*  D0
;       ED = 1110 1101
;            D7*  D6*  D5* /D4*  D3*  D2* /D1*  D0
;       DD = 1101 1101
;            D7*  D6* /D5*  D4*  D3*  D2* /D1*  D0
;       FD = 1111 1101
;            D7*  D6*  D5*  D4*  D3*  D2* /D1*  D0

/CBED       =(/D6_3    * /D5* /D4*      /D2*  D1* /D0_7
             +/D6_3    *  D5* /D4*       D2* /D1* /D0_7)
            * /M1* R_CBED

/DDFD       =(/D6_3    * /D5*  D4*      D2* /D1* /D0_7
             +/D6_3    *  D5*  D4*      D2* /D1* /D0_7)
            * /M1* R_CBED

/MCSYNC     = /RD              ;MCSYNC LOW
            + /WR              ;WHEN READ ,WRITE OR INTAK
            + /M1 * /IORQ
```

วงจรอีมูเลชันซีพียู Z-80 (รูป 4.9)

อีมูเลชันซีพียู Z-80 ต่อกับอุปกรณ์รอบนอก ซึ่งมีส่วนที่ต่อไปยังฮาร์ดแวร์ของระบบเป้าหมาย และส่วนที่ต่อกับวงจรควบคุมพอร์ตของคอนโทรลซีพียูที่ใช้ตรวจสอบ และควบคุมการทำงานของอีมูเลชันซีพียู

อุปกรณ์ที่ทำหน้าที่หลักในการส่งสัญญาณควบคุม คือ พอร์ต C ของไอซี 8255 BUS เป็นพอร์ตเอาต์พุต มีสัญญาณดังนี้

E_INT เมื่อสัญญาณเป็น 0 อนุญาตให้สัญญาณอินเตอร์รัพต์จากระบบเป้าหมายต่อไปยังอีมูเลชันซีพียูได้

NEXT เป็นสัญญาณนาฬิกาของฟลิปฟลอปที่ผลิตสัญญาณ WAIT และ BUSRQ สำหรับควบคุมอีมูเลชันซีพียู ทำงานที่ขอบขาขึ้น

F_BUSRQ เมื่อเป็น 0 จะทำให้เกิดสัญญาณ BUSRQ ที่อีมูเลชันซีพียู

E_RESET เมื่อเป็น 0 อนุญาตให้สัญญาณรีเซตจากระบบเป้าหมายต่อไปยังอีมูเลชันซีพียูได้

F_WAIT เมื่อเป็น 1 ทำให้เกิดสัญญาณ WAIT ที่อีมูเลชันซีพียู

E_NMI เมื่อเป็น 0 อนุญาตให้สัญญาณอินเตอร์รัพต์แบบ Non-Maskable จากระบบเป้าหมายต่อไปยังอีมูเลชันซีพียูได้

EMU เมื่อเป็น 1 อีมูเลชันซีพียูอยู่ในโมดอีมูเลชันเมื่อเป็น 0 อีมูเลชันซีพียูอยู่ในโมดอินเทอร์โรเกชัน

ENABLE เมื่อเป็น 1 ในโมดอีมูเลชันอนุญาตให้ใช้ BUSRQ จากระบบเป้าหมาย ในโมดอินเทอร์โรเกชันอนุญาตให้อีมูเลชันซีพียูต่อกับหน่วยความจำของผู้ใช้

พอร์ตอื่นๆ ได้แก่

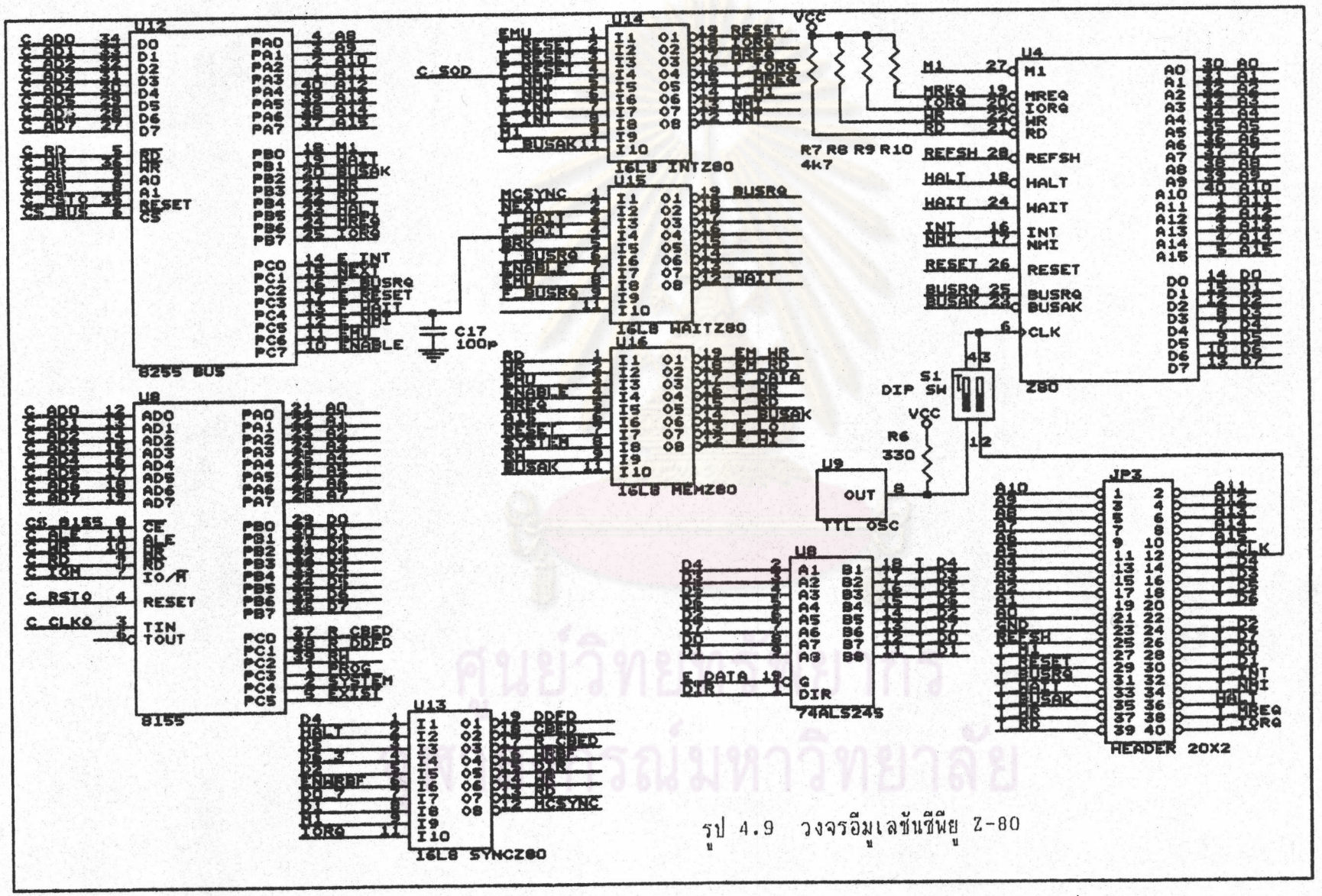
พอร์ต B ของ 8255 BUS ใช้อ่านสถานะของอีมูเลชันซีพียู

พอร์ต A ของ 8255 BUS ใช้อ่านแอดเดรสไบต์สูง

พอร์ต A ของ 8155 ใช้อ่านแอดเดรสไบต์ต่ำ

พอร์ต B ของ 8155 ใช้อ่านบัสข้อมูล และใส่คำสั่งให้อีมูเลชันซีพียู

PAL16L8 WAITZ80 ทำหน้าที่แทนฟลิปฟลอปสำหรับสัญญาณ WAIT และ BUSRQ มีสัญญาณ WAIT1, WAIT2, BUSRQ1, BUSRQ2 เป็นเกตที่ใช้ต่อกันภายในเพื่อสร้างเป็นฟลิปฟลอป มีสมการดังนี้



รูป 4.9 วงจรอีมูเลชันซีพียู Z-80

```
;WAIT1 WITH WAIT2 -> D-FF OUT HIGH AT RISING EDGE OF 'NEXT'
;                    (DON'T CARE F_WAIT, RUN THAT MACHINE CYCLE)
;                    CLEAR OUTPUT WHEN MCSYNC HIGH AND F_WAIT OR BRK
;                    (END OF MACHINE CYCLE WAIT MAY ACTIVE AGAIN)
 WAIT1    = (/NEXT* VCC              ;CLK LO OUT FOLLOW DATA HIGH
           + NEXT* WAIT1)            ;CLK HI LATCH OLD OUTPUT
          */( MCSYNC* F_WAIT         ;RESET FROM F_WAIT
           + MCSYNC* /BRK)           ;       OR BRK

 WAIT2    = ( NEXT* WAIT1            ;CLK HI OUT FOLLOW DATA WAIT1
           +/NEXT* WAIT2)            ;CLK LO LATCH OLD OUTPUT
          */( MCSYNC* F_WAIT         ;RESET FROM F_WAIT
           + MCSYNC* /BRK)           ;       OR BRK

/WAIT     = /WAIT2                   ;WAIT ACTIVE FROM D-FF
          + /T_WAIT* EMU             ;OR TARGET WAIT IN EMULATION
          + /T_WAIT* /EMU* ENABLE    ;OR TARGET WAIT WHEN ACCESS MEM/IO

;BUSRQ1 WITH BUSRQ2 -> D-FF OUT HIGH AT RISING EDGE OF 'NEXT'
;                    (DON'T CARE F_BUSRQ, TO NEXT MACHINE CYCLE)
;                    CLEAR OUTPUT WHEN MCSYNC LOW AND F_BUSRQ
;                    (IN MACHINE CYCLE BUSRQ MAY ACTIVE AGAIN)
 BUSRQ1   = (/NEXT* VCC              ;CLK LO OUT FOLLOW DATA HIGH
           + NEXT* BUSRQ1)           ;CLK HI LATCH OLD OUTPUT
          */(/MCSYNC* /F_BUSRQ)      ;RESET FROM F_BUSRQ

 BUSRQ2   = ( NEXT* BUSRQ1           ;CLK HI OUT FOLLOW DATA BUSRQ1
           +/NEXT* BUSRQ2)           ;CLK LO LATCH OLD OUTPUT
          */(/MCSYNC* /F_BUSRQ)      ;RESET FROM F_BUSRQ
```

```
/BUSRQ    = /BUSRQ2                    ;BUSRQ ACTIVE FROM D-FF
          +  /T_BUSRQ* EMU* ENABLE     ;OR TARGET WHEN ENABLE BUSRQ
```

จากการทดลองให้อีมูเลชันซีพียูทำงานตามเวลาจริงจะเกิดสัญญาณ WAIT ขึ้นทำให้หยุดการทำงานโดยไม่ถูกต้องต้องต่อตัวเก็บประจุเพื่อลดสัญญาณรบกวนที่ขา F_WAIT จึงหมดปัญหา

สัญญาณที่ทำให้เกิดการ WAIT นอกจาก F_WAIT ยังมีสัญญาณ BRK ที่มาจากวงจรควบคุมการหยุดด้วย สำหรับ MCSYNC เมื่อเป็น 0 แสดงว่ามีการอ่านเขียนบัสตามที่อธิบายในเรื่องการควบคุมการทำงานของซีพียู

PAL 16L8 INTZ80 ใช้สร้างสัญญาณที่จะอินเตอร์รัพต์ซีพียู โดยมีที่มาจากการควบคุมให้สัญญาณต่อกับระบบเป้าหมายหรือไม่ และในช่วงอินเทอร์โรเกชันจะไม่ต่อสัญญาณเหล่านี้กับระบบเป้าหมาย สมการของสัญญาณมีดังนี้

```
/RESET    = /T_RESET* /E_RESET* EMU ;ENABLE RESET FROM TARGET IN EMU
          + /F_RESET                ;FORCE RESET BY CONTROL CPU
                                    ;ENABLE NMI FROM TARGET WHEN
 NMI      = /E_NMI* EMU* T_NMI      ;ENABLE IN EMU
          + /(/E_NMI* EMU)* NMI     ;LATCH WHEN NOT ENABLE
          + /T_RESET* /E_RESET      ;NMI NOT ACTIVE AFTER RESET
          + /F_RESET

/INT      = /E_INT* /T_INT* EMU     ;ENABLE INT FROM TARGET WHEN ENABLE

/T_M1     = /M1* EMU                ;ENABLE M1 TO TARGET IN EMU
```

อุปกรณ์รอบนอกที่ออกแบบมาสำหรับ Z-80 จะใช้สัญญาณ M1 ในการควบคุมการทำงานภายในด้วย

```
/T_MREQ  = /MREQ
 T_MREQ.TRST = T_BUSAK

/T_IORQ  = /IORQ
 T_IORQ.TRST = T_BUSAK
```

BUSAK เกิดขึ้นทุกครั้งที่ใช้ BUSRQ ในการควบคุมการทำงานจึงต้องป้องกันไม่ให้ต่อไปยังระบบเป้าหมายในโมดอินเทอร์โรเกชัน

MREQ และ IOREQ ถูกควบคุมด้วย BUSAK จึงต้องต่อความต้านทานให้ระดับสัญญาณเป็น 1 ผ่านบัฟเฟอร์ที่จะเป็นสถานะอิมพีแดนซ์สูงเมื่อเกิด BUSAK ที่ระบบเป้าหมาย

ในรูปจะพบสัญญาณ F_RESET ต่อมาจาก C_SOD ซึ่งสั่งงานจากคอนโทรลซีพียู เมื่อคอนโทรลซีพียูถูกรีเซตสัญญาณนี้จะเป็น 0 ทำให้อีมูเลชันซีพียูถูกรีเซตด้วย สำหรับสัญญาณ NMI เมื่อไม่ต่อกับระบบเป้าหมายจะให้คงสถานะเดิม เพราะการเปลี่ยนสถานะอาจทำให้เกิดอินเตอร์รัพต์ขึ้นได้

สัญญาณออกจากอีมูเลชันซีพียูที่ต่อไปยังระบบเป้าหมายจะถูกควบคุมให้ปรากฏที่ระบบเป้าหมาย ในช่วงที่มีการใช้งานสัญญาณนั้นจริงๆ เท่านั้น เพราะหลายสัญญาณถูกสร้างขึ้นจากการควบคุมการทำงานของซีพียู และอาจก่อให้เกิดการทำงานผิดพลาดที่ระบบเป้าหมายได้ ได้แก่สัญญาณ

```
/DIR  =   /RD
        + /M1 * /IORQ
```

ควบคุมบัฟเฟอร์ของข้อมูลให้มีทิศทางเข้าเมื่อซีพียูต้องการอ่าน

สัญญาณที่ต่อโดยตรงระหว่างระบบเป้าหมายกับอีมูเลชันซีพียู ได้แก่ บัสแอดเดรส, REFSH และ HALT

สัญญาณนาฬิกาที่ให้กับอีมูเลชันซีพียูสามารถเลือกได้จากออสซิลเตอร์ภายในความถี่ 4 เมกกะเฮิรตซ์ หรือจากระบบภายนอกด้วย DIP SW กรณีที่ภายนอกไม่มีสัญญาณแต่ต้องการใช้ให้ต่อสวิทช์เลือกภายนอกและภายใน

สำหรับสัญญาณ T_RD และ T_WR จะอธิบายในเรื่องหน่วยความจำอีมูเลชัน

วงจรอีมูเลชันซีพียู 8085 (รูป 4.10)

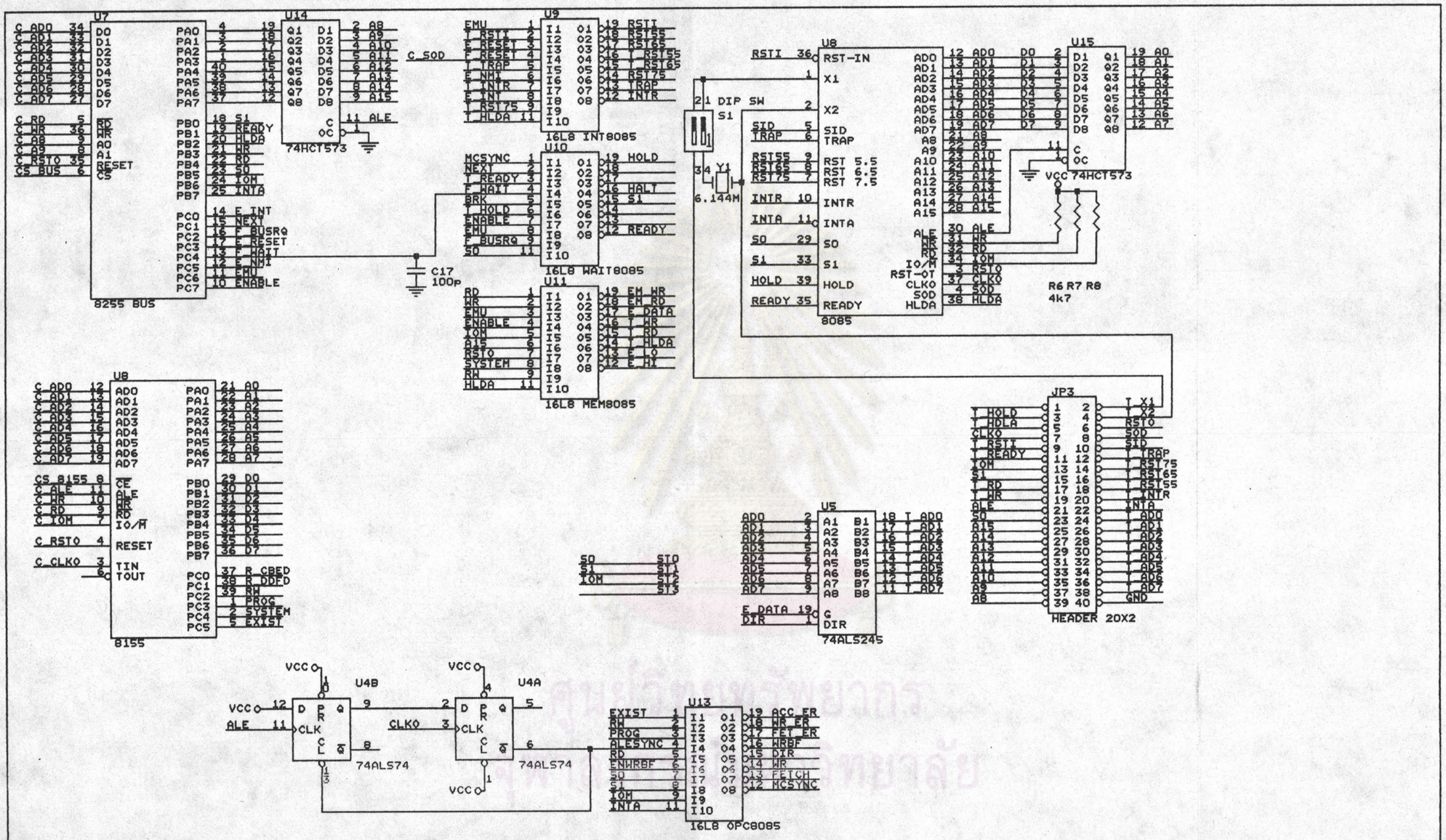
มีรูปแบบการทำงานเช่นเดียวกับอีมูเลชันซีพียู Z-80 ที่ได้กล่าวมาแล้วแตกต่างเพียงรายละเอียดเล็กน้อย เช่นระดับหรือช่วงเวลาของสัญญาณ และการมีบัสแอดเดรสไบต์ต่ำมัลติเพลกซ์กับบัสข้อมูล ทำให้พอร์ตที่อ่านค่าแอดเดรสต้องผ่านวงจรแลตช์ รวมทั้งแอดเดรสไบต์สูงต้องผ่านวงจรแลตช์ด้วยเพื่อให้ทราบค่าแอดเดรสต่อจากที่โปรแกรมพบคำสั่ง HLT เพราะหลังจาก HALT บัสแอดเดรสจะมีสถานะอิมพีแดนซ์สูง

MCSYNC ของ 8085 ที่สร้างจากสัญญาณการอ่านเขียน และตอบรับอินเตอร์รัพต์สามารถใช้งานได้ดีในการสร้างสัญญาณ HOLD/READY แต่ในการกำหนดการหยุด เมื่อโปรแกรมหยุดการทำงานด้วยคำสั่ง HLT จะต้องการสัญญาณเกิดจาก ALE เราจึงนำ ALE มาสร้างสัญญาณ ALESYNC เพื่อรวมกับ MCSYNC แบบปรกติ รูปแบบวงจรสร้าง ALESYNC ใช้ฟลิปฟลอป 2 ตัว ตัวแรกสร้างเอาต์พุตจากขอบขาขึ้นของ ALE อีกตัวจัดสัญญาณจากตัวแรกให้ตรงกับขอบขาขึ้นของ CLKO ผลที่ได้จะเกิดสัญญาณเริ่มในช่วงที่สามารถใช้แลตช์แอดเดรสได้และจบในช่วงที่สัญญาณอ่านเขียนปรากฏแล้ววงจรนี้เป็นวงจรเดียวกับที่ใช้สร้างสัญญาณ READY เพื่อเพิ่มช่วงเวลาการอ่านเขียนอีก 1 คาบสัญญาณนาฬิกา

สมการที่ใช้ใน PAL ที่ควบคุมการทำงานมีดังนี้

```
;WAIT1 WITH WAIT2 -> D-FF OUT HIGH AT RISING EDGE OF 'NEXT'
;                    (DON'T CARE F_WAIT, RUN THAT MACHINE CYCLE)
;                    CLEAR OUTPUT WHEN MCSYNC HIGH AND F_WAIT OR BRK
;                    (END OF MACHINE CYCLE WAIT MAY ACTIVE AGAIN)
 WAIT1   = (/NEXT* VCC              ;CLK LO OUT FOLLOW DATA HIGH
          + NEXT* WAIT1)            ;CLK HI LATCH OLD OUTPUT
         */( MCSYNC* F_WAIT         ;RESET FROM F_WAIT
          + MCSYNC* /BRK)           ;       OR BRK

 WAIT2   = ( NEXT* WAIT1            ;CLK HI OUT FOLLOW DATA WAIT1
          +/NEXT* WAIT2)            ;CLK LO LATCH OLD OUTPUT
         */( MCSYNC* F_WAIT         ;RESET FROM F_WAIT
          + MCSYNC* /BRK)           ;       OR BRK
```



รูป 4.10 วงจรอีมูเลชันซีพียู 8085

```
/READY    = /WAIT2                    ;WAIT ACTIVE FROM D-FF
          + /T_READY* EMU             ;OR TARGET WAIT IN EMULATION
          + /T_READY* /EMU* ENABLE    ;OR TARGET WAIT WHEN ACCESS MEM/IO

;BUSRQ1 WITH BUSRQ2 -> D-FF OUT HIGH AT RISING EDGE OF 'NEXT'
;                      (DON'T CARE F_BUSRQ, TO NEXT MACHINE CYCLE)
;                      CLEAR OUTPUT WHEN MCSYNC LOW AND F_BUSRQ
;                      (IN MACHINE CYCLE BUSRQ MAY ACTIVE AGAIN)
 BUSRQ1   = (/NEXT* VCC               ;CLK LO OUT FOLLOW DATA HIGH
            + NEXT* BUSRQ1)           ;CLK HI LATCH OLD OUTPUT
          */(/MCSYNC* /F_BUSRQ)       ;RESET FROM F_BUSRQ

 BUSRQ2   = ( NEXT* BUSRQ1            ;CLK HI OUT FOLLOW DATA BUSRQ1
            +/NEXT* BUSRQ2)           ;CLK LO LATCH OLD OUTPUT
          */(/MCSYNC* /F_BUSRQ)       ;RESET FROM F_BUSRQ

 HOLD     = /BUSRQ2                   ;BUSRQ ACTIVE FROM D-FF
          + T_HOLD* EMU* ENABLE       ;OR TARGET WHEN ENABLE BUSRQ

/DIR      = /RD                       ;DATA BUS DIRECTION = IN
          + /INTA                     ;WHEN READ OR INTA

/MCSYNC   = /RD                       ;MCSYNC LOW
          + /WR                       ;WHEN READ ,WRITE OR INTAK
          + /INTA
          + /ALESYNC                  ;OR NEXT CLOCK CYCLE FROM ALE
```

```
;BUSAK AT TARGET ACTIVE WHEN
 T_HLDA     =  HLDA* EMU* ENABLE          ;HLDA IN EMU AND ENABLE(BUSRQ)

/RSTI       = /T_RSTI* /E_RESET* EMU     ;ENABLE RESET FROM TARGET IN EMU
            + /F_RESET                   ;FORCE RESET BY CONTROL CPU
                                         ;ENABLE TRAP FROM TARGET
/TRAP       = /E_NMI* EMU* /T_TRAP       ;WHEN ENABLE IN EMU
            + /(/E_NMI* EMU)* /TRAP      ;LATCH WHEN NOT ENABLE
            + /T_RSTI* /E_RESET          ;TRAP NOT ACTIVE AFTER RESET
            + /F_RESET

 INTR       = /E_INT* T_INTR* EMU        ;ENABLE INT FROM TARGET WHEN ENABLE
 RST55      = /E_INT* T_RST55* EMU       ;ENABLE INT FROM TARGET WHEN ENABLE
 RST65      = /E_INT* T_RST65* EMU       ;ENABLE INT FROM TARGET WHEN ENABLE
                                         ;ENABLE INT FROM TARGET
/RST75      = /E_INT* EMU* /T_RST75      ;WHEN ENABLE IN EMU
            + /(/E_INT* EMU)* /RST75     ;LATCH WHEN NOT ENABLE
            + /T_RSTI* /E_RESET          ;INT NOT ACTIVE AFTER RESET
            + /F_RESET
```

สัญญาณ TRAP และ RST75 ขณะที่ไปต่อกับระบบเป้าหมายต้องคงสถานะเดิมไว้ เพราะการเปลี่ยนสถานะอาจทำให้เกิดอินเตอร์รัพต์ได้

สัญญาณที่อีมูเลชันซีพียูต่อกับระบบเป้าหมายโดยตรง ได้แก่ บัสแอดเดรสไบต์สูง, ALE, IOM, RSTO, CLKO, SOD, SO, S1, INTA และ SID

สัญญาณนาฬิกาภายในระบบบัสสร้างจากคริสตัล 6.144 เมกกะเฮิรตซ์ สามารถต่อสายสัญญาณนาฬิกาจากภายนอกได้โดยปรับที่ DIP SW แต่ถ้าอุปกรณ์สร้างสัญญาณนาฬิกาภายนอกเป็นคริสตัลธรรมดาที่ต้องต่อกับขา X1 X2 ของ 8085 โดยตรง ต้องใช้สายสั้นที่สุด

วงจรหน่วยความจำอีมูเลชัน (รูป 4.11)

อินเซอร์กิตอีมูเลเตอร์ที่ออกแบบขึ้นมีหน่วยความจำอีมูเลชันขนาด 64 กิโลไบต์ มีวงจรเลือกใช้และกำหนดลักษณะของหน่วยความจำด้วยคำสั่งได้ช่วงละ 2 กิโลไบต์ จึงต้องมีวงจรถอดรหัสแอดเดรสแบบโปรแกรมได้เป็นส่วนประกอบ โดยวงจรนี้ต้องทำงานให้ตัวเลือกหน่วยความจำและลักษณะของหน่วยความจำออกมาหลังจากซีพียูส่งแอดเดรสให้ไปแล้ว และต้องทำงานได้เร็วพอที่จะได้สัญญาณออกมาก่อนที่ซีพียูจะส่งสัญญาณอ่านเขียนหน่วยความจำออกมา ในที่นี้เลือกหน่วยความจำชนิดทีทีแอล ขนาด 16x4 บิต เบอร์ 74LS189 2 ตัว การต่อกับระบบไมโครโปรเซสเซอร์จึงต้องใช้บัฟเฟอร์ชนิดซีมอส ในที่นี้เป็นแบบอินเวอร์เตอร์เบอร์ 74HCT540 เพราะมีเวลาหน่วงน้อยกว่าบัฟเฟอร์แบบธรรมดา และลดความสับสนในการใช้หน่วยความจำสำหรับถอดรหัสที่มีเอาต์พุตแบบอินเวอร์เตอร์ด้วย

การเชื่อมต่อวงจรส่วนนี้ให้สัญญาณแอดเดรสของ 74LS189 มาจากอีมูเลชันซีพียู และข้อมูลเข้ามาจากคอนโทรลซีพียู และมีพอร์ต C ของ 8155 เป็นตัวอ่านค่าที่ได้จากหน่วยความจำถอดรหัสเพื่อรายงานผลการจัดหน่วยความจำที่ผู้ใช้กำหนดสัญญาณแอดเดรส และข้อมูลเข้าออกของหน่วยความจำถอดรหัสทั้ง 2 ตัวต่อเข้าด้วยกัน เลือกการใช้ทีละตัวด้วยสัญญาณ E_LO และ E_HI จาก PAL การเขียนหน่วยความจำทำด้วยสัญญาณ CS_MAP

รูปแบบของข้อมูล 4 บิต ที่เก็บไว้มีความหมายดังนี้

RW เป็น 0 คือ หน่วยความจำอ่านอย่างเดียว

เป็น 1 คือ หน่วยความจำที่เขียนได้

PROG เป็น 0 คือ หน่วยความจำที่ไม่ใช่โปรแกรมห้ามซีพียูอ่านเป็นโปรแกรม

เป็น 1 คือ หน่วยความจำที่เป็นโปรแกรมหรือข้อมูล

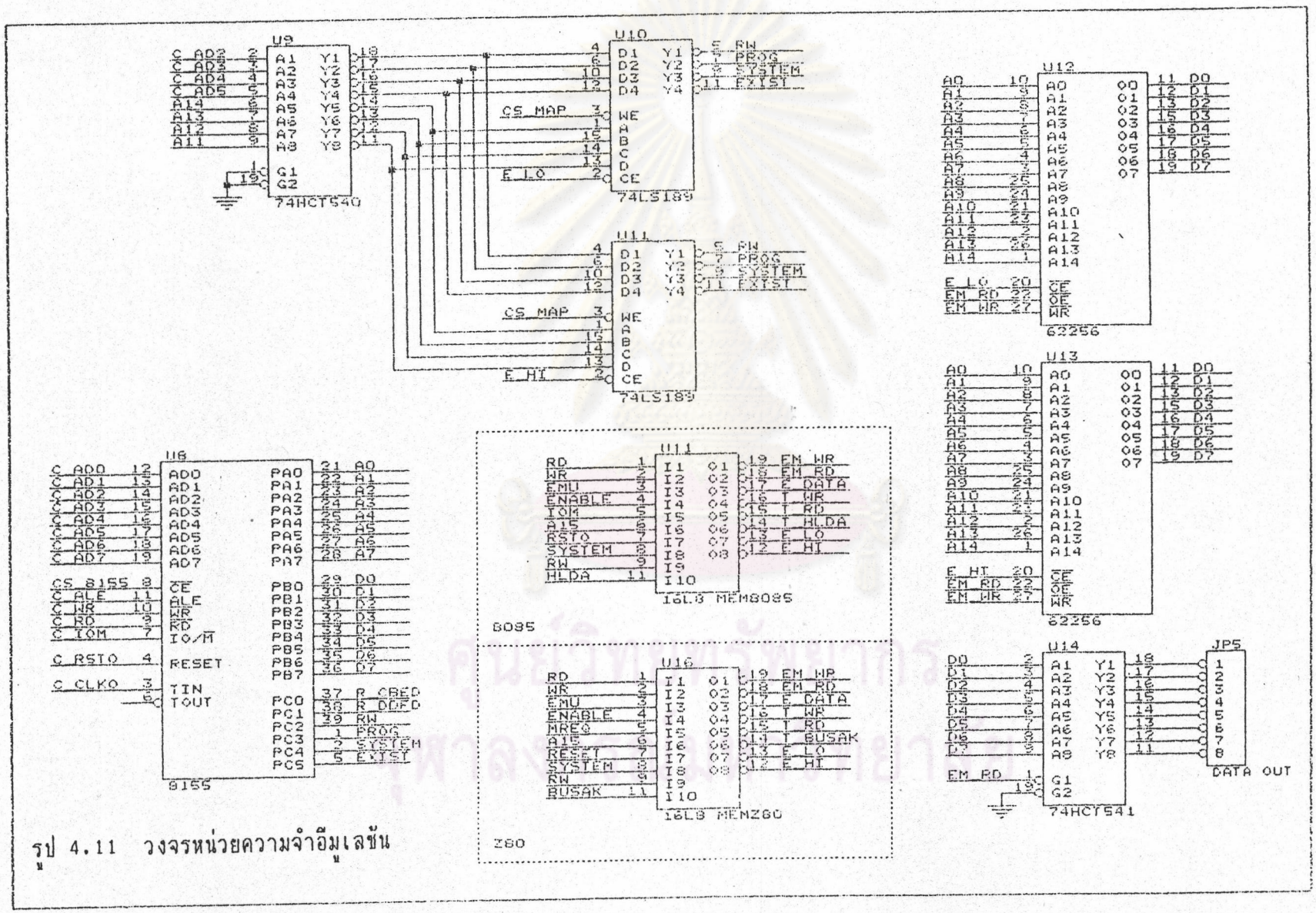
SYSTEM เป็น 0 คือ เลือกใช้หน่วยความจำจากระบบเป้าหมาย

เป็น 1 คือ เลือกใช้หน่วยความจำอีมูเลชัน

EXIST เป็น 0 คือ ไม่มีหน่วยความจำในช่วงแอดเดรสนี้

เป็น 1 คือ มีหน่วยความจำในช่วงแอดเดรสนี้

สัญญาณที่เลือกใช้หน่วยความจำจริงๆ คือ SYSTEM และ RW ที่นำไปใช้ร่วมกับสัญญาณจากอีมูเลชันซีพียูและวงจรควบคุม เพื่อสร้างสัญญาณอ่านเขียนส่งไปยังระบบเป้าหมาย และหน่วยความจำอีมูเลชัน



รูป 4.11 วงจรหน่วยความจำอีมูเลชัน

การต่อหน่วยความจำอีมูเลชัน มีลักษณะคล้ายกับหน่วยความจำปรกติ คือ มีบัสแอดเดรส ข้อมูลต่อกับบัสของซีพียู (สำหรับ 8085 แอดเดรสไบต์ต่ำจะต่อผ่านแลตช์) ในวงจรนี้จะใช้การแยกหน่วยความจำบนระบบเป้าหมายด้วยการใช้บัฟเฟอร์ของบัสข้อมูล กันข้อมูลจากภายนอกขณะอ่านภายใน และกันสัญญาณอ่านภายในขณะอ่านจากภายนอก

การเลือกใช้หน่วยความจำอีมูเลชัน

ขณะอ่าน มีสัญญาณอ่านมาที่หน่วยความจำอีมูเลชัน และมีสัญญาณอ่านส่งไประบบเป้าหมายด้วยแต่บัฟเฟอร์ข้อมูลของอีมูเลชันซีพียูไม่ทำงาน ข้อมูลจากภายนอกจึงไม่เข้ามา

ขณะเขียน ข้อมูลจากซีพียูส่งไปยังหน่วยความจำทั้งระบบเป้าหมายและหน่วยความจำอีมูเลชัน ถ้า RW เป็น 1 จะเขียนข้อมูลทั้ง 2 ไปพร้อมๆกัน และวิธีเขียนขณะมีการเลือกใช้หน่วยความจำภายนอกก็จะได้ผลเช่นเดียวกัน

การอ่านหน่วยความจำจากระบบเป้าหมาย จะส่งสัญญาณอ่านมาที่ระบบเป้าหมายและยอมให้ข้อมูลผ่านบัฟเฟอร์เข้ามาได้ แต่ไม่ส่งสัญญาณอ่านมาที่หน่วยความจำอีมูเลชัน

สมการ PAL ของวงจรสร้างสัญญาณอ่านเขียนหน่วยความจำของ Z-80 และ 8085 มีลักษณะต่างกันเล็กน้อย เพราะสัญญาณอ่านเขียนแบบเดียวกัน เช่น MREQ ของ Z-80 เหมือนกับ IOM ของ 8085 ในขณะอ่านเขียนหน่วยความจำ แต่ BUSAK ของ Z-80 ทำงานที่ลอจิก 0 แต่ HLDA ของ 8085 ทำงานที่ลอจิก 1 และข้อแตกต่างที่สำคัญ คือ 8085 มีการส่งแอดเดรสออกทางบัสข้อมูลด้วย บัฟเฟอร์บัสข้อมูลของอีมูเลชันซีพียูที่ต่อไปยังระบบเป้าหมายต้องยอมให้แอดเดรสออกไปได้ทุกแมชชีนไซเคิล

สมการ PAL ของ Z-80

```
;READ AT TARGET ACTIVE WHEN
/T_RD      =  BUSAK*                          ;BUSAK NOT ACTIVE AND
           ( /RD  * /MREQ * /EMU* ENABLE      ;MEM_RD IN INTERR AND ENABLE
           + /RD  * /MREQ *  EMU              ;MEM_RD IN EMU
           + /RD  *  MREQ )                   ;IO_RD
 T_RD.TRST =  T_BUSAK                         ;BUSAK AT TARGET NOT ACTIVE
```

```
;WRITE AT TARGET ACTIVE WHEN
/T_WR       =  BUSAK*                          ;BUSAK NOT ACTIVE AND
            ( /WR  * /MREQ * /EMU* ENABLE      ;MEM_WR IN INTERR AND ENABLE
            + /WR  * /MREQ *  EMU* RW          ;MEM_WR IN EMU AND RW
            + /WR  *  MREQ )                   ;IO_WR
 T_WR.TRST =  T_BUSAK                          ;BUSAK AT TARGET NOT ACTIVE

;DISABLE DATA BUS BETWEEN CPU AND TARGET WHEN
 E_DATA     = /RD  * /MREQ * SYSTEM            ;READ MEMORY FROM SYSTEM
            + /EMU* /ENABLE                    ;INTERR AND NOT ENABLE(MEM)
            + /BUSAK                           ;BUSAK ACTIVE
            + /RESET                           ;RESET ACTIVE

;READ AT EMULATION MEMORY ACTIVE WHEN
/EM_RD      =  BUSAK*                          ;BUSAK NOT ACTIVE AND
            ( /RD  * /MREQ * SYSTEM* /EMU
             * ENABLE                          ;RD_SYS IN INTERR AND ENABLE
            + /RD  * /MREQ * SYSTEM*  EMU);RD_SYS IN EMU

;WRITE AT EMULATION MEMORY ACTIVE WHEN
/EM_WR      =  BUSAK*                          ;BUSAK NOT ACTIVE AND
            ( /WR  * /MREQ * /EMU* ENABLE      ;WR IN INTERR AND ENABLE(MEM)
            + /WR  * /MREQ *  EMU* RW )        ;WR IN EMU AND RW

/E_LO       = /A15                             ;ENABLE LOW WHEN A15 = 0

/E_HI       =  A15                             ;ENABLE HIGH WHEN A15 = 1
```

สมการ PAL ของ 8085

```
;READ AT TARGET ACTIVE WHEN
/T_RD       = /HLDA*                          ;BUSAK NOT ACTIVE AND
            ( /RD  * /IOM  * /EMU* ENABLE     ;MEM_RD IN INTERR AND ENABLE
            + /RD  * /IOM  *  EMU             ;MEM_RD IN EMU
            + /RD  *  IOM  )                  ;IO_RD
 T_RD.TRST = /T_HLDA                          ;BUSAK AT TARGET NOT ACTIVE

;WRITE AT TARGET ACTIVE WHEN
/T_WR       = /HLDA*                          ;BUSAK NOT ACTIVE AND
            ( /WR  * /IOM  * /EMU* ENABLE     ;MEM_WR IN INTERR AND ENABLE
            + /WR  * /IOM  *  EMU* RW         ;MEM_WR IN EMU AND RW
            + /WR  *  IOM  )                  ;IO_WR
 T_WR.TRST = /T_HLDA                          ;BUSAK AT TARGET NOT ACTIVE

;DISABLE DATA BUS BETWEEN CPU AND TARGET WHEN
 E_DATA     = /RD  * /IOM  * SYSTEM           ;READ MEMORY FROM SYSTEM
            + /RD  * /EMU* /ENABLE            ;INTERR READ BUT NOT ENABLE
            + /WR  * /EMU* /ENABLE            ;INTERR WRITE BUT NOT ENABLE
            +  HLDA                           ;BUSAK ACTIVE
            +  RSTO                           ;RESET ACTIVE

;READ AT EMULATION MEMORY ACTIVE WHEN
/EM_RD      = /HLDA*                          ;BUSAK NOT ACTIVE AND
            ( /RD  * /IOM  * SYSTEM* /EMU
             * ENABLE                         ;RD_SYS IN INTERR AND ENABLE
            + /RD  * /IOM  * SYSTEM*  EMU);RD_SYS IN EMU
```

```
;WRITE AT EMULATION MEMORY ACTIVE WHEN
/EM_WR      = /HLDA*                          ;BUSAK NOT ACTIVE AND
            ( /WR  * /IOM  * /EMU* ENABLE ;WR IN INTERR AND ENABLE
            + /WR  * /IOM  *  EMU* RW )   ;WR IN EMU AND RW

/E_LO       = /A15                            ;ENABLE LOW WHEN A15 = 0

/E_HI       =  A15                            ;ENABLE HIGH WHEN A15 = 1
```

ส่วนสำคัญอีกส่วนที่มีประโยชน์สำหรับการใช้ Z-80 กับอุปกรณ์เชื่อมต่อมาตรฐานของ Z-80 ที่ต้องอ่านรหัสคำสั่งไปพร้อมกับ Z-80 ด้วย แต่ขณะอ่านบัสของอินเซอร์กิตอีมูเลเตอร์ควรจะเป็นอินพุตเนื่อให้เหมือนกับซีพียูจริงๆการนำข้อมูลออกไปภายนอก ในวงจรนี้จะใช้บัฟเฟอร์ที่อนุญาตให้สัญญาณจากหน่วยความจำอีมูเลชันออกไปภายนอกได้ทางขั้วต่อ DATA OUT ขณะอินเซอร์กิตอีมูเลเตอร์อ่านหน่วยความจำอีมูเลชัน

<u>วงจรควบคุมการหยุด</u> (รูป 4.12)

การหยุดของอินเซอร์กิตอีมูเลเตอร์ที่ออกแบบมี 7 แบบ แต่ละแบบมีที่มาดังนี้

1. การกำหนดแอดเดรสของโปรแกรม

8255 BRK มีพอร์ต A กำหนดแอดเดรสไบต์สูง พอร์ต B กำหนดแอดเดรสไบต์ต่ำ พอร์ต C ส่งสัญญาณ E_BP เป็น 1 ทำให้มีการหยุดจากวงจรนี้ ค่าแอดเดรสที่กำหนดจะเปรียบเทียบ 74HC688 2 ตัว แต่ละตัวมีสัญญาณที่แสดงว่าเป็นช่วงอ่านโปรแกรม เป็นตัวควบคุมการทำงาน

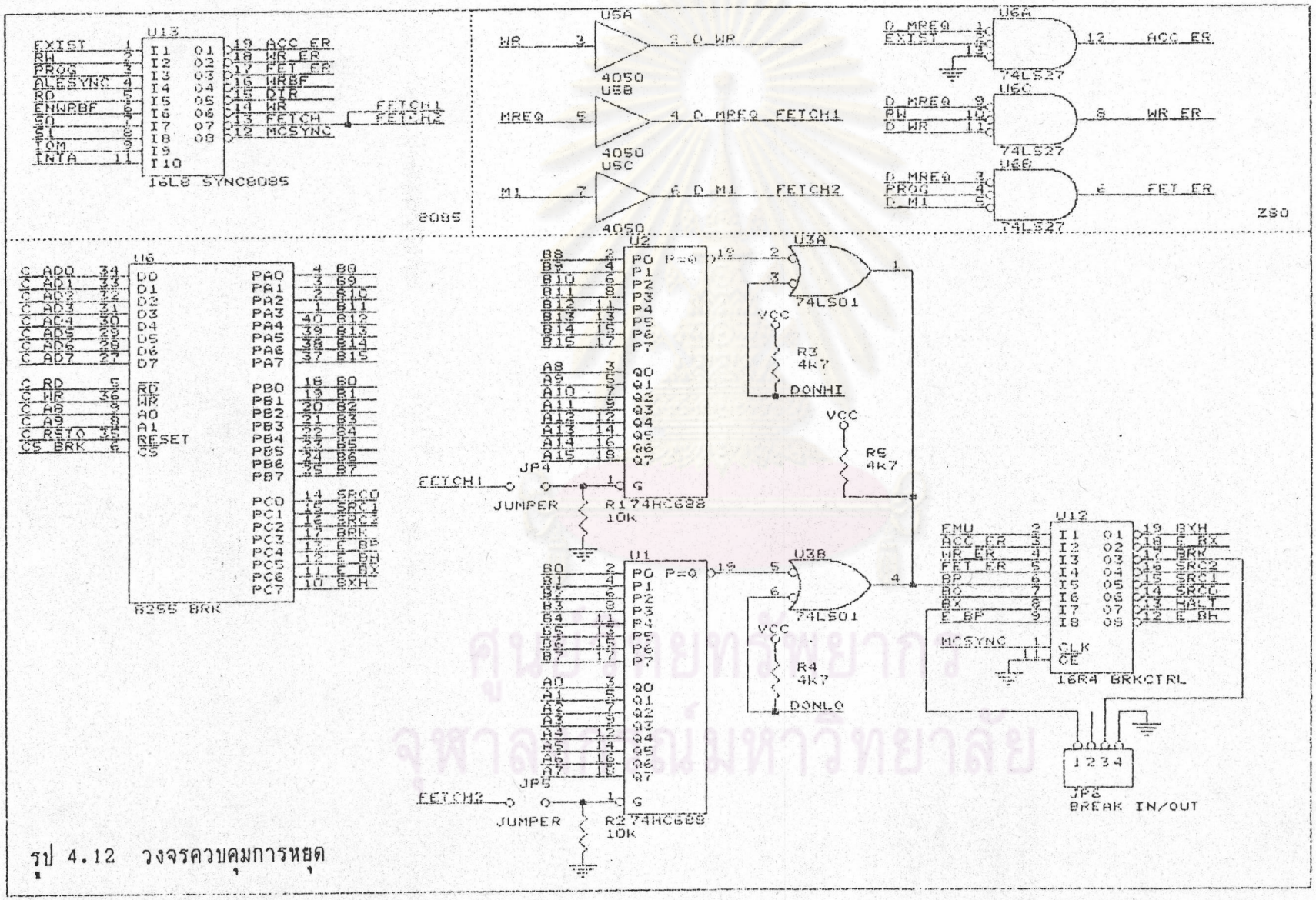
สัญญาณ FETCH จากวงจร 8085

```
/FETCH  =  /IOM * S1 * SO
```

และสัญญาณ D_MREQ และ D_M1 จาก Z-80

สัญญาณจากไอซีเปรียบเทียบต้องได้ค่าตรงกับที่กำหนดทั้งคู่จึงจะได้ BP เป็น 1 ที่ PAL 16R4 BRKCTRL

2. หยุดเมื่อบัฟเฟอร์ติดตามการทำงานตามเวลาจริงเต็ม โดยจะได้สัญญาณ BO เป็น 0



รูป 4.12 วงจรควบคุมการหยุด

3. หยุดเมื่อสัญญาณภายนอกตรงกับลอจิกที่กำหนด กำหนดให้ใช้เมื่อ 8255 BRK พอร์ต C สัญญาณ E_BX เป็น 1ลอจิกที่กำหนดจะตรงกับสัญญาณ BXH

4. หยุดเมื่อเขียนข้อมูลในหน่วยความจำที่กำหนดให้เป็นแบบอ่านอย่างเดียว ใช้สัญญาณจาก WR_ER

สำหรับ 8085

WR_ER = /RW * /IOM * /S1 * SO

สำหรับ Z-80 จากวงจรเขียนได้เป็น

WR_ER = /RW * /D_MREQ * /D_WR

5. หยุดเมื่ออ่านคำสั่งจากหน่วยความจำที่เก็บเฉพาะข้อมูล ใช้สัญญาณจาก FET_ER

สำหรับ 8085

FET_ER = /PROG * /IOM * S1 * SO

สำหรับ Z-80 จากวงจรเขียนได้เป็น

FET_ER = /PROG * /D_MREQ * /D_M1

6. หยุดเมื่ออ่านเขียนหน่วยความจำในแอดเดรสที่กำหนดว่าไม่มีหน่วยความจำ

สำหรับ 8085

ACC_ER = /EXIST * /IOM

สำหรับ Z-80 จากวงจรเขียนได้เป็น

ACC_ER = /EXIST * /D_MREQ

7. หยุดเมื่อโปรแกรม HALT

ตั้งโดยกำหนด 8255 BRK พอร์ต C สัญญาณ E_BH เป็น 1 และจะหยุดเมื่อ HALT เป็น 0

สำหรับ 8085

/HALT = /SO * /S1

วงจรที่ทำหน้าที่รวมสัญญาณเหล่านี้ส่งไปให้วงจรควบคุมการทำงานของซีพียู คือ 16L4 BRKCTRL มีสัญญาณออก 4 เส้น คือ

BRK เป็น 0 ทำให้วงจรควบคุมสร้างสัญญาณ WAIT

SRC2	SRC1	SRC0	บอกแหล่งที่มาของสัญญาณที่ทำให้หยุด
0	0	0	โปรแกรม HALT
0	0	1	อ่านเขียนบริเวณที่ไม่มีหน่วยความจำ
0	1	0	อ่านคำสั่งจากข้อมูล
0	1	1	เขียนหน่วยความจำอ่านอย่างเดียว
1	0	0	มีสัญญาณจากภายนอก
1	0	1	บัฟเฟอร์เต็ม
1	1	0	จากการกำหนดจุดหยุด

สัญญาณออกที่ได้จาก SCR2 SCR1 SCR0 เป็นลักษณะของ priority encoder คือ จัดความสำคัญให้กับสัญญาณที่ทำให้เกิดการหยุด สัญญาณเข้าที่มีความสำคัญสูงสุด คือ HALT และรองลงมาจนถึงการกำหนดจุดหยุดตามลำดับ

สมการทั้งหมดแสดงได้ดังนี้

```
;BRK LOW TO FORCE EMULATION CPU TO WAIT
/BRK    := E_BH* /HALT          ;7 :BREAK WHEN CPU HALT
         + ACC_ER* EMU          ;6 :ACCESS MEMORY ERROR IN EMU
         + FET_ER* EMU          ;5 :FETCH ERROR IN EMU
         + WR_ER* EMU           ;4 :WRITE ERROR IN EMU
         + E_BX*  BXH*  BX      ;3 :BREAK WHEN EXTERNAL HIGH
         + E_BX* /BXH* /BX      ;3 :BREAK WHEN EXTERNAL LOW
         + /BO                  ;2 :BREAK FROM OTHER BOARD
         + E_BP*  BP            ;1 :BREAK FROM ADDRESS COMPARATOR
```

```
;CONTROL CPU READ STATUS OF BREAK
; SRC2 SRC1 SRC0
;  0    0    0   7 BREAK WHEN HALT
;  0    0    1   6 ACCESS NON-EXISTENT
;  0    1    0   5 FETCH OPCODE FROM DATA MEMORY
;  0    1    1   4 WRITE TO READ ONLY MEMORY
;  1    0    0   3 BREAK EXTERNAL
;  1    0    1   2 BREAK FROM OTHER BOARD
;  1    1    0   1 BREAK POINT 1
;  1    1    1   0 INVALID
;  PRIORITY ENCODER
; /SRC0 = 1 * /2 * /4 * /6
;       + 3      * /4 * /6
;       + 5           * /6
;       + 7
; /SRC1 = 2      * /4 * /5
;       + 3      * /4 * /5
;       + 6
;       + 7
; /SRC2 = 4
;       + 5
;       + 6
;       + 7

/SRC0   := (E_BP*  BP)                    * /(/BO)* /(WR_ER)* /(ACC_ER)
         + (E_BX*  BXH*  BX + E_BX* /BXH* /BX)  * /(WR_ER)* /(ACC_ER)
         + (FET_ER)                                        * /(ACC_ER)
         + (E_BH* /HALT)
```

```
/SRC1   := (/BO)                                    * /(WR_ER)* /(FET_ER)
         + (E_BX*  BXH*  BX + E_BX* /BXH* /BX)   * /(WR_ER)* /(FET_ER)
         + (ACC_ER)
         + (E_BH* /HALT)
/SRC2   := (WR_ER)
         + (FET_ER)
         + (ACC_ER)
         + (E_BH* /HALT)
```

<u>วงจรติดตามการทำงานในเวลาจริง</u> (รูป 4.13)

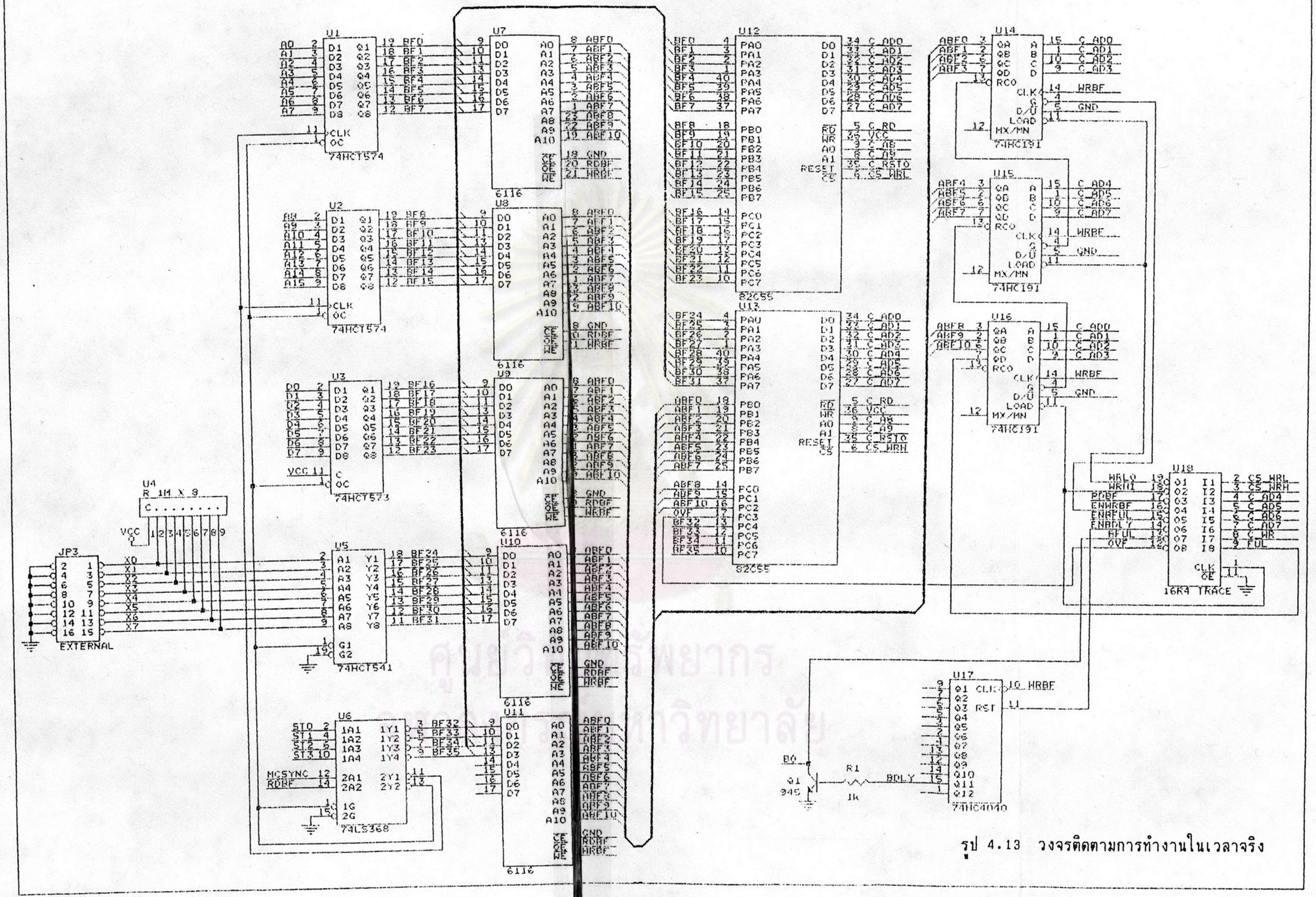
ในขณะอีมูเลชันซีพียูทำงานตามเวลาจริงในโมดอีมูเลชัน วงจรนี้จะวัดสัญญาณจากซีพียูเพื่อติดตามว่าซีพียูทำงานตามโปรแกรมของผู้ใช้ได้ค่าแอดเดรสข้อมูล และสถานะในแต่ละแมชชีนไซเคิลเป็นอย่างไร ค่าที่วัดได้จะถูกเก็บไว้ในหน่วยความจำบัฟเฟอร์ขนาด 2048 x 40 บิต แบ่งเป็นบัฟเฟอร์สำหรับเก็บแอดเดรสของซีพียู 16 บิต ข้อมูล 8 บิต สถานะ 4 บิต และมีสายต่อวัดสัญญาณจากวงจรภายนอกมาเก็บในบัฟเฟอร์นี้ด้วยขนาด 8 บิต

หน่วยความจำส่วนนี้ ประกอบจากหน่วยความจำแรมเบอร์ 6116 ที่มีขนาด 2048 x 8 บิต จำนวน 5 ตัว ต่อแอดเดรสเข้าด้วยกันและใช้งานอ่านเขียนพร้อมๆกัน โดยต่อสัญญาณขาที่มีชื่อ CE ของ 6116 ทุกตัวกับ GND

ในการอ่านสถานะจากหน่วยความจำบัฟเฟอร์จะใช้ไอซี 8255 2 ตัว เมื่อรีเซตทุกพอร์ตเป็นอินพุต ในวงจรนี้เราไม่ต่อสัญญาณ WR กับสัญญาณที่ใช้สำหรับเขียนข้อมูลแต่ให้ลอจิก 1 ตลอดเวลา จึงโปรแกรมให้ 8255 เป็นพอร์ตเอาต์พุตไม่ได้ พอร์ต 8255 ใช้เป็นอินพุตสำหรับอ่านสถานะทุกอย่างของวงจรนี้ เพื่อส่งไปให้คอนโทรลซีพียูรับรู้ สัญญาณที่ใช้เลือกการทำงานของ 8255 คือ CS_WRH และ CS_WRL

วงจรที่ใช้สร้างแอดเดรสสำหรับกำหนดตำแหน่งข้อมูลในหน่วยความจำบัฟเฟอร์ คือ วงจรนับ 4 บิต เบอร์ 74HC191 จำนวน 3 ตัว วงจรนับนี้สามารถกำหนดค่าเริ่มต้นได้ เราจึงต่อในลักษณะเอาต์พุตพอร์ตของคอนโทรลซีพียู จำนวน 2 พอร์ต มีสัญญาณเลือกสำหรับเขียนข้อมูลต่อเข้าที่ขา LOAD ได้สัญญาณมาจาก WRLO และ WRHI ซึ่งมีสมการดังนี้

```
/WRLO  =  /C_WR * /CS_WRL
/WRHI  =  /C_WR * /CS_WRH
```



รูป 4.13 วงจรติดตามการทำงานในเวลาจริง

นั่นคือมีหมายเลขพอร์ตซ้ำกับที่ใช้เลือก 8255 แต่ 8255 เป็นพอร์ตอินพุต พอร์ตที่ใช้สัญญาณนี้เลือกเป็นพอร์ตเอาต์พุต

แอดเดรสของบัฟเฟอร์มี 11 เส้น ใช้ชื่อ ABF0 ถึง ABF1 การตั้งแอดเดรสด้วยการโปรแกรมค่าทำให้วงจรนับ สัญญาณ WRLO ใช้ตั้ง ABF0 ถึง ABF7 สัญญาณ WRHI ใช้ตั้ง ABF8 ถึง ABF10 และใช้สร้างสัญญาณ อีก 4 สัญญาณ คือ

RDBF เป็น 0 จะให้ข้อมูลจากหน่วยความจำบัฟเฟอร์ออกมาให้ 8255 อ่านได้

ENWRBF เป็น 0 ส่งไปให้ส่วนควบคุมการทำงานของอีมูเลชันซีพียูที่สร้าง MCSYNC ทำให้เกิดสัญญาณ WRBF สำหรับเขียนข้อมูลที่วัดได้ในบัฟเฟอร์เมื่อจบแมชชีนไซเคิล

ENBFUL เป็น 0 จะทำให้เกิดสัญญาณ BO เมื่อหน่วยความจำบัฟเฟอร์เต็ม

ENBDLY เป็น 0 จะทำให้เกิดสัญญาณ BO หลังจากหน่วงเวลาไป 1024 แมชชีนไซเคิล โดยใช้วงจรนับ 74HC4040

สัญญาณ BO ส่งไปให้วงจรควบคุมการหยุดตามที่ได้อธิบายแล้ว

สัญญาณอื่นๆ ที่สร้างในวงจรนี้ ได้แก่

FUL คือ ABF11 เป็น 1 แสดงว่าวงจรนับทำงานจนครบทุกแอดเดรสแล้ว บัฟเฟอร์เต็ม

```
OVF  =   FUL
       + OVF * WRLO
```

ทำหน้าที่ฟลิปฟลอปที่ค้างสัญญาณ FUL ไว้ เพื่อให้ตรวจสอบได้ว่าบัฟเฟอร์เต็มและฟลิปฟลอปนี้จะถูกรีเซต เมื่อมีการตั้งค่าการนับใหม่

WRBF ในช่วงที่ ENWRBF เป็น 0 WRBF จะเหมือน MCSYNC ทำหน้าที่สร้างสัญญาณเขียน หน่วยความจำบัฟเฟอร์เมื่อจบแมชชีนไซเคิล และเพิ่มค่าแอดเดรสของบัฟเฟอร์ขึ้น เพื่อรอบันทึกค่าที่วัดในแมชชีนไซเคิลต่อไป

สำหรับ Z-80

```
/WRBF  =   (/RD + /WR + /M1 * /IORQ + /HALT)
         * /ENWRBF
```

สำหรับ 8085

```
/WRBF  =   (/RD + /WR + /INTA + /ALESYNC)
         * /ENWRBF
```

BFUL เป็นสัญญาณออกชนิดคอลเลคเตอร์เปิดสร้างโดยใช้การควบคุมเอาต์พุตแบบ 3 สถานะของ PAL

```
BFUL.TRST  =  FUL  *  /ENBFUL

BFUL       =  GND
```

BDLY เป็น 1 เมื่อนับครบ 1024 แมชชีนไซเคิล หลังจาก ENBDLY เป็น 0

BO เป็นสัญญาณออกชนิดคอลเลคเตอร์เปิด ที่เกิดจากสัญญาณ BFUL หรือ BDLY ทำงานที่ลอจิก 0

สัญญาณสถานะของซีพียูที่จะบันทึก คือ ST0 ถึง ST3 สำหรับ Z-80 ดูจากรูป 4.10 สัญญาณเหล่านี้ต่อผ่านไอซีอินเวอร์เตอร์ 74LS368 ก่อนเก็บในหน่วยความจำบัฟเฟอร์ BF32 ถึง BF35 มีความหมายดังนี้

สำหรับ Z-80

BF35(HALT)	BF34(M1)	BF33(MREQ)	BF32(WR)	สถานะ
0	1	1	0	อ่านออพโค้ด
0	0	1	0	อ่านหน่วยความจำ
0	0	1	1	เขียนหน่วยความจำ
0	0	0	0	อ่านอินพุต
0	0	0	1	เขียนเอาต์พุต
0	1	0	0	ตอบรับอินเตอร์รัพต์
1	x	x	x	HALT

สำหรับ 8085

BF35(HALT)	BF34(M1)	BF33(MREQ)	BF32(WR)	สถานะ
x	1	0	0	อ่านออพโค้ด
x	1	0	1	อ่านหน่วยความจำ
x	1	1	0	เขียนหน่วยความจำ
x	0	0	1	อ่านอินพุต
x	0	1	0	เขียนเอาต์พุต
x	0	0	0	ตอบรับอินเตอร์รัพต์
x	x	1	1	HALT

สัญญาณที่วัดมาจากภายนอกจะผ่านไอซี 74HCT541 ก่อนเข้าสู่บัสข้อมูลของหน่วยความจำบัฟเฟอร์

สัญญาณจากบัสข้อมูลของซีพียูผ่านไอซี 74HCT573 โดยไม่ใช้งานแลตช์ ยอมให้สัญญาณผ่านโดยมีเวลาหน่วงมากกว่า 74HCT541 เพื่อหน่วงเวลาข้อมูลให้นานพอ สำหรับการเขียนหน่วยความจำบัฟเฟอร์

สัญญาณแอดเดรสจะผ่านฟลิปฟลอป เพื่อแลตช์สัญญาณที่ขอบขาลงของ MCSYNC ซึ่งเป็นช่วงที่แอดเดรสยังไม่เปลี่ยนแปลง การบันทึกจะได้ค่าถูกต้องกับทุกสถานะ เพราะแอดเดรสในสถานะ HALT จะเปลี่ยนแปลงก่อนถึงขอบขาขึ้นของ MCSYNC

ไอซีที่ต่อสัญญาณที่วัดไปยังหน่วยความจำบัฟเฟอร์ จะมีเอาต์พุตแบบ 3 สถานะเพื่อให้เอาต์พุตเป็นอิมพีแดนซ์สูงในช่วงที่อ่านข้อมูลจากหน่วยความจำบัฟเฟอร์ (RDBF = 0)

คำสั่งเกี่ยวกับการติดตามการทำงานในเวลาจริง มี 3 คำสั่ง คือ

Trace Backward

เก็บข้อมูลตั้งแต่เริ่มทำงานจนถึงหยุดทำงาน ถ้าข้อมูลเต็มจะเก็บทับข้อมูลในช่วงแรก ข้อมูลที่ได้ คือการติดตามการทำงานในช่วง 2048 แมชชีนไซเคิลสุดท้ายก่อนจะหยุดตามจุดหยุดที่ผู้ใช้ตั้งไว้

ในคำสั่งนี้ต้องการฮาร์ดแวร์ที่จะบันทึกว่าข้อมูลเต็มจนทับช่วงแรกหรือไม่ เพื่อหาตำแหน่งเริ่มต้นบัฟเฟอร์ที่จะแสดงผล (สัญญาณ OVF)

Trace Center

เก็บข้อมูลตั้งแต่เริ่มทำงานจนพบจุดหยุดแล้วเก็บข้อมูลต่ออีก 1024 แมชชีนไซเคิล จึงหยุดการทำงาน ถ้าข้อมูลเต็มจะเก็บทับข้อมูลในช่วงแรก ข้อมูลที่ได้คือการติดตามการทำงานในช่วง 2048 แมชชีนไซเคิลสุดท้าย

คำสั่งนี้ต้องการฮาร์ดแวร์ในการสร้างจุดหยุดโดยนับแมชชีนไซเคิล 1024 ไซเคิล (74HC4040 และสัญญาณ ENBDLY BDLY) และดูว่าข้อมูลมีการทับตำแหน่งช่วงแรกหรือไม่ เพื่อหาตำแหน่งเริ่มต้นเวลาแสดงผล (สัญญาณ OVF)

Trace Forward

ทำงานตามปรกติโดยไม่มีการบันทึกจนพบจุดหยุดจึงเริ่มบันทึก แล้วทำงานต่อจนบัฟเฟอร์เต็มแล้วหยุดการทำงาน ข้อมูลที่ได้ คือ 2048 ไซเคิล หลังพบจุดหยุด

คำสั่งนี้ต้องการฮาร์ดแวร์ในการสร้างจุดหยุดเมื่อบัฟเฟอร์เต็ม (สัญญาณ BFUL)