บทที่ 7

การประยุกต์ใช้งาน

7.1 การประยุกต์ใช้งานระบบพัฒนา

ระบบพัฒนาไมโครโปรเซสเซอร์ระดับบอร์ดนี้ สามารถนำไปประยุกต์ใช้งานได้หลายด้าน ทั้งในด้านการศึกษา ด้านงานวิจัยและพัฒนา และในด้านการใช้งานจริง ในด้านการศึกษาผู้ใช้อาจเพิ่งเริ่มศึกษาเกี่ยวกับระบบไมโครโปรเซสเซอร์ ผู้ใช้สามารถใช้ระบบพัฒนานี้ศึกษาการทำงานของซีพียู การอ่านเขียนหน่วยความจำ การอ่านเขียนพอร์ตไอ/โอ และการทำงานของแต่ละคำสั่งในภาษาแอสเซมบลี นอกจากนี้ผู้ใช้ยังสามารถฝึกหัดเขียนโปรแกรมได้โดยไม่ต้องเข้าไปยุ่งเกี่ยวกับฮาร์ดแวร์ของระบบไมโครโปรเซสเซอร์เลย ในด้านงานวิจัยและพัฒนาผู้ใช้สามารถออกแบบและพัฒนาซอฟต์แวร์ได้ โดยเสียเวลาในการออกแบบและดีบักฮาร์ดแวร์น้อยมาก ทั้งยังมีอุปกรณ์และโปรแกรมต่าง ๆ ที่ช่วยในการดีบักซอฟต์แวร์อยู่ในระบบด้วย จึงเป็นการช่วยลดเวลาที่ใช้ในการวิจัยและพัฒนาระบบไมโครโปรเซสเซอร์ลง และยังสามารถนำระบบพัฒนาไมโครโปรเซสเซอร์ระดับบอร์ดนี้ไปใช้ในงานจริง หลังจากวิจัยพัฒนาเรียบร้อยแล้วด้วย

การใช้งานระบบพัฒนาไมโครโปรเซสเซอร์ระดับบอร์ดนี้ สามารถแบ่งออกเป็น 2 ขั้นตอนใหญ่ ๆ คือ

7.1.1 ขั้นตอนการออกแบบพัฒนาฮาร์ดแวร์

- เริ่มจากผู้ใช้ออกแบบวงจรโดยอาศัยบอร์ดต่าง ๆ ที่มีอยู่ในระบบ หรือใช้ STD บอร์ดที่มีขายทั่วไปซึ่งมีให้เลือกมากมายเกือบทุกฟังก์ชัน ดังจะแสดงตัวอย่างรายการ STD บอร์ดของบริษัท Versalogic ได้ดังรูปที่ 7.1 หากวงจรมีส่วนประยุกต์ใช้งานพิเศษซึ่งไม่มีฮาร์ดแวร์ที่เหมาะสม ผู้ใช้สามารถต่อวงจรเพิ่มเติมบนบอร์ดสนับสนุนด้วยวิธีการ Wirewrap เพื่อใช้ในการทดสอบ และทดลองกับฮาร์ดแวร์ ซอฟต์แวร์ก่อน หลังจากนั้นค่อยนำวงจรที่ได้ไปทำแผ่นวงจรที่หลัง

- ตรวจดูว่าตั้งค่าตัวเลือกต่าง ๆ ถูกต้องหรือไม่ ไม่ว่าจะเป็นJumperเลือกขนาด/แอดเดรสของหน่วยความจำผู้ใช้ ตัวสลับขาสัญญาณของหน่วยความจำรอม/แรมบนบอร์ดซีพียูในหัวข้อ 4.4.2 ตัวเลือกแอดเดรสพอร์ตไอ/โอและตัวต่อ Jumper ต่าง ๆ ในบทที่ 6

- นำบอร์ดมาเสียบเข้ากับคอนเนกเตอร์บนบอร์ดแม่ซึ่งอยู่ติดอยู่กับแร็ก

- สามารถใช้บอร์ดอีมูเลชันช่วยในการตรวจหาความผิดพลาดของฮาร์ดแวร์ด้วยคำสั่งต่าง ๆ เช่น อ่านค่าจากพอร์ต("I") หรือเขียนค่าไปที่พอร์ตไอ/โอ ("O") เป็นต้น เพื่อตรวจสอบการเชื่อมต่อของสายสัญญาณแอดเดรส สายสัญญาณข้อมูล และสายสัญญาณควบคุม

- หลังจากประกอบวงจร และตรวจสอบฮาร์ดแวร์เรียบร้อยแล้วสามารถออกแบบเขียนโปรแกรมทดสอบส่วนต่างๆ และพัฒนาไปสู่โปรแกรมควบคุมระบบที่ประยุกต์ใช้งาน

VersaLogic CORP.

STD Bus Price List

			1-9	10-24	25-49
PARALLEL I/O	VL-7601A	32-Line Input + 32-Line Output TTL I/O	150	135	127
	VL-7602A	64-Line TTL Output	150	135	127
	VL-7603A	64-Line TTL Input	150	135	127
	VL-7604A	64-Line Configurable TTL I/O	160	144	135
	VL-7614	64-Line Configurable TTL I/O	180	162	152
INDUSTRIAL I/O	VL-7507	24-Line Opto 22 Type Interface	160	144	135
	VL-7508	48-Line Opto 22 Type Interface	215	194	182
	VL-IPI-1	8-Channel Opto-Isolated AC/DC Input	175	163	150
	VL-IPI-2	8-Line Relay Output	225	209	193
	VL-IPI-3	8-Line AC Solid State Relay Output	295	274	252
	VL-IPI-4	4 IPI-1 Inputs, 4 IPI-2 Outputs	200	186	171
	VL-IPI-5	4 IPI-1 Inputs, 4 IPI-3 Outputs	275	256	235
ANALOG I/O	VL-1225	16S/8D-Chan. 10-Bit Analog In., 2-Chan. 8-Bit Out.	405	373	350
	VL-1226	16S/8D-Channel, 10-Bit Analog Input	330	304	285
	VL-1260	16S/8D (Opt. 32/16) Channel, 12-Bit, Analog Input	480	442	415
	VL-1262	4-Channel, 12-Bit, Analog Output	380	350	329
	VL-AIN-1a	8-Channel, 12-Bit, Analog Input	265	246	227
	VL-AIN-1b	Above with Current Loop Inputs	275	256	235
	VL-AOUT-1	4-Channel, 10-Bit, Analog Output	225	209	193
MEMORY	VL-7709a	64-256K RAM/ROM Board, Unpopulated	135	122	114
	VL-7709a16	64K RAM/ROM Board with 16K RAM	Contact Factory		
	VL-7709a32	256K RAM/ROM Board with 32K RAM	Contact Factory		
	VL-7709a64e	256K RAM/ROM Board with 64K RAM	Contact Factory		
	VL-7709a128	256K RAM/ROM Board with 128K RAM	Contact Factory		
	VL-7709a256	256K RAM/ROM Board with 256K RAM	Contact Factory		
	VL-7709b	64-256K Battery-Backed RAM/ROM Board, Unpopulated	180	162	152
	VL-7709b32	256K Battery-Backed RAM/ROM Board with 32K RAM	Contact Factory		
	VL-7709b64	256K Battery-Backed RAM/ROM Board with 64K RAM	Contact Factory		
	VL-7709b256	256K Battery-Backed RAM/ROM Board with 256K RAM	Contact Factory		
OTHER FUNCTIONS	VL-VID-80	80-Character, 24-Line Video Controller	225	209	193
	VL-VID-64	64-Character, 24-Line Video Controller	225	209	193
	VL-7914	Prototyping Card with Bus Interface	80	72	68

รูปที่ 7.1 แสดงตัวอย่างรายการ STD บอร์ดของบริษัท Versalogic

จากที่กล่าวมาสรุปขั้นตอนการออกแบบฮาร์ดแวร์ได้ดังรูปที่ 7.2

รูปที่ 7.2 แสดงขั้นตอนการออกแบบฮาร์ดแวร์

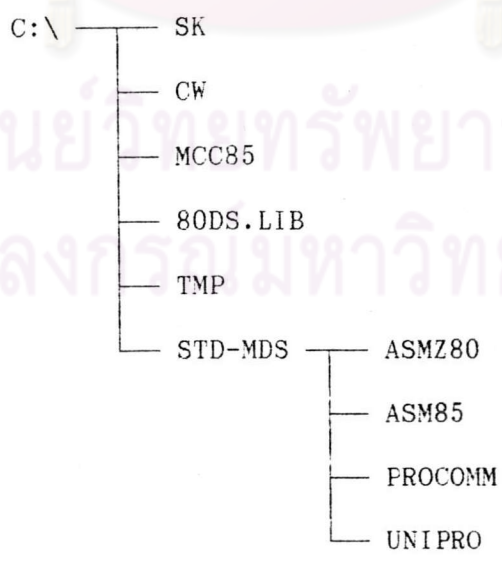
7.1.2 ขั้นตอนการออกแบบพัฒนาซอฟต์แวร์

ในงานวิจัยครั้งนี้ ได้ทำระบบซอฟต์แวร์สำหรับการพัฒนาโปรแกรมขึ้นในฮาร์ดดิสค์ของเครื่องไมโครคอมพิวเตอร์ IBM PC ระบบซอฟต์แวร์ที่สร้างขึ้นประกอบด้วยโปรแกรมต่าง ๆ หลาย Package โปรแกรมส่วนใหญ่ถูกจัดอยู่ในไดเรกตอรีย่อย ๆ ในไดเรกตอรีใหญ่ชื่อ "STD-MDS" ซึ่งภายในไดเรกตอรี STD-MDS นี้จะมีไฟล์ QEdit ที่เป็นโปรแกรมเอดิเตอร์และไดเรกตอรีย่อยดังนี้

- ASMZ80 ซึ่งมี Software Package ของแอสเซมเบลอร์ Z-80
- ASM85 ซึ่งมี Software Package ของแอสเซมเบลอร์ 8085
- PROCOMM ซึ่งมีโปรแกรมติดต่อกับบอร์ดอีมูเลชัน

และ UNIPRO ซึ่งมีโปรแกรมใช้ในการโปรแกรมอีพรอม

ส่วนบาง Package ที่เป็นโปรแกรมใช้งานทั่วไปจะอยู่ในไดเรกตอรีย่อยของไดเรกตอรีรากได้แก่ SideKick อยู่ในไดเรกตอรีชื่อ "SK",CuWriterอยู่ในไดเรกตอรีชื่อ "CW" และ Package ของภาษา C85 ซึ่งมีขนาดใหญ่มากจะอยู่ในไดเรกตอรีชื่อ "MCC85" ส่วน Software Package ของ PL/M85 จำเป็นต้องอยู่ในไดเรกตอรีชื่อ "80DS.LIB"และต้องมีไดเรกตอรีชื่อ "TMP" อยู่ที่ไดเรกตอรีรากด้วย ดังแสดงระบบซอฟต์แวร์ที่สร้างขึ้นได้ในรูปที่ 7.3 ส่วนไฟล์ต่าง ๆ ในแต่ละไดเรกตอรีแสดงไว้ในภาคผนวก ค



รูปที่ 7.3 แสดงระบบซอฟต์แวร์ที่สร้างขึ้น

ในบทนี้เราจะอธิบายการใช้งานระบบอย่างคร่าว ๆ ส่วนการใช้งาน Package ต่าง ๆ ผู้ใช้ต้องไปศึกษารายละเอียดจากคู่มือต่อไปนี้

QEdit -> SEMWARE. QEdit ADVAVCED : SEMWARE ,1990.

SideKick -> Borland. SideKick Version 1.56A :Borland,1985.

CuWriter -> จุฬาลงกรณ์มหาวิทยาลัย. CuWriter Version 1.52,1991.

MCC85 -> Microtech Research.Paragon MCC85/MCCZ80/MCC180 Version 1.5C Cross Compiler ,1987.[24]

PL/M85 -> 80/DS Expert-PL/M Microprocessor Program Development Tools : Fiber and Gordon ,1986.

ASM85/ASMZ80 -> Microtech Research.Paragon MCC85/MCCZ80/MCC180 Version 1.5C Cross Compiler ,1987.[24]

PROCOMM -> Bob Campbell. Mastering PROCOMM Plus V 1.1B : Sybex ,1988.

UNIPRO -> XelTex. Universal Programming Manual Book,1990.

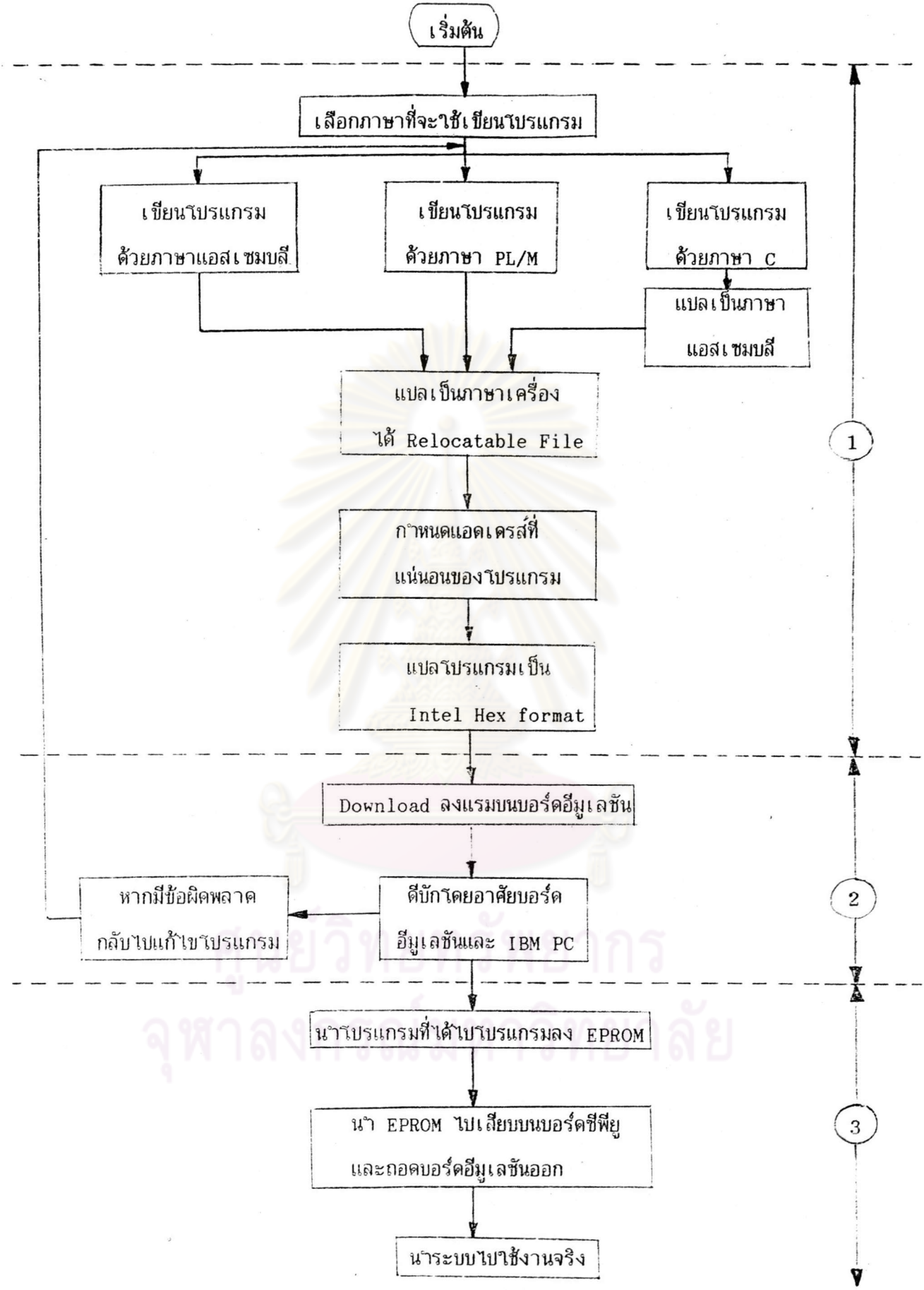
การออกแบบและพัฒนาซอฟต์แวร์ของผู้ใช้แบ่งได้เป็น 3 ช่วงใหญ่ ๆ คือ

ช่วงที่ 1. ออกแบบและเขียนโปรแกรม

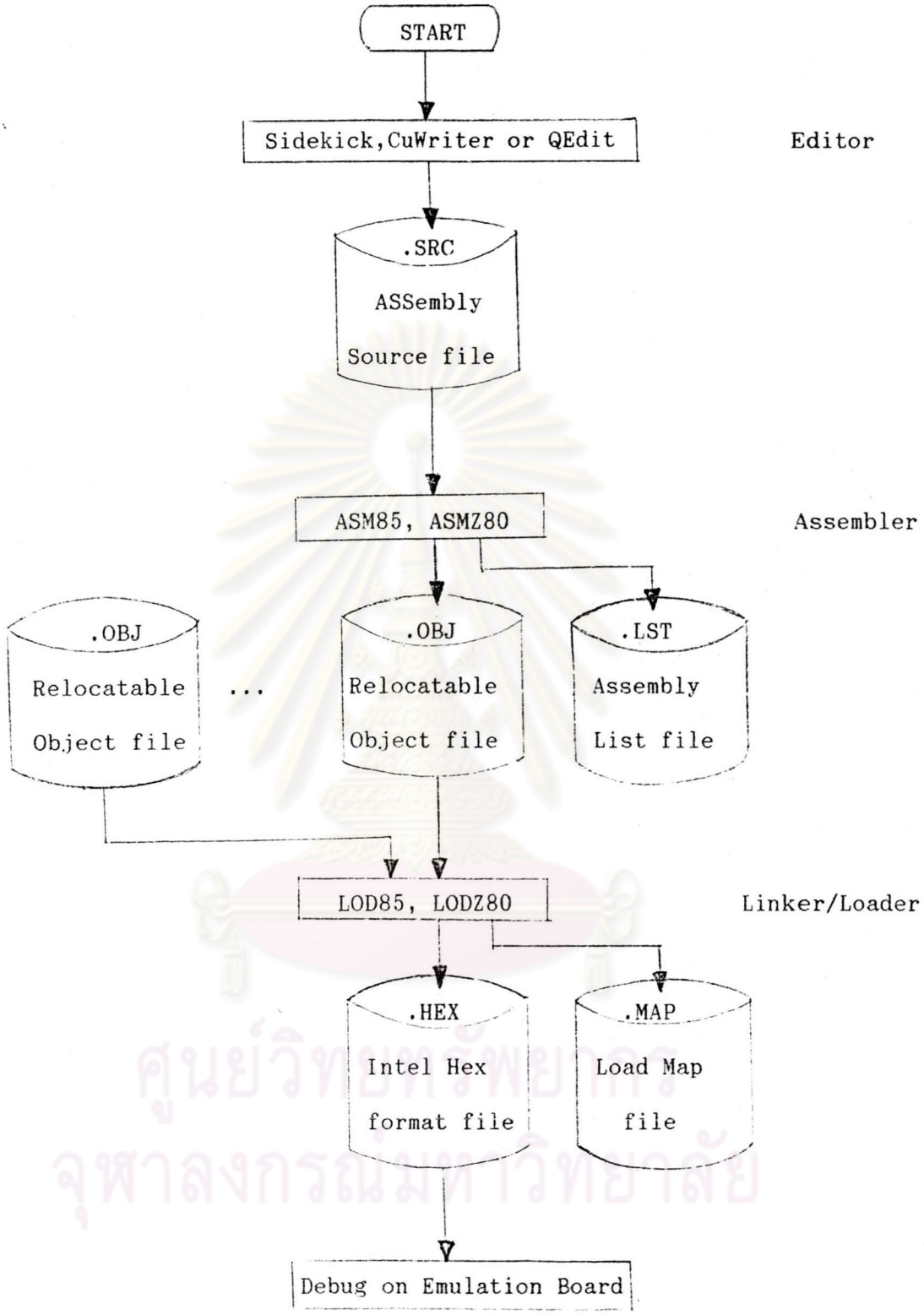
ช่วงที่ 2. ทดสอบและแก้ไขโปรแกรม

ช่วงที่ 3. นำโปรแกรมที่พัฒนาเรียบร้อยแล้วไปใช้งาน

ซึ่งเราสามารถแสดงขั้นตอนการออกแบบพัฒนาซอฟต์แวร์ได้ดังรูปที่ 7.4

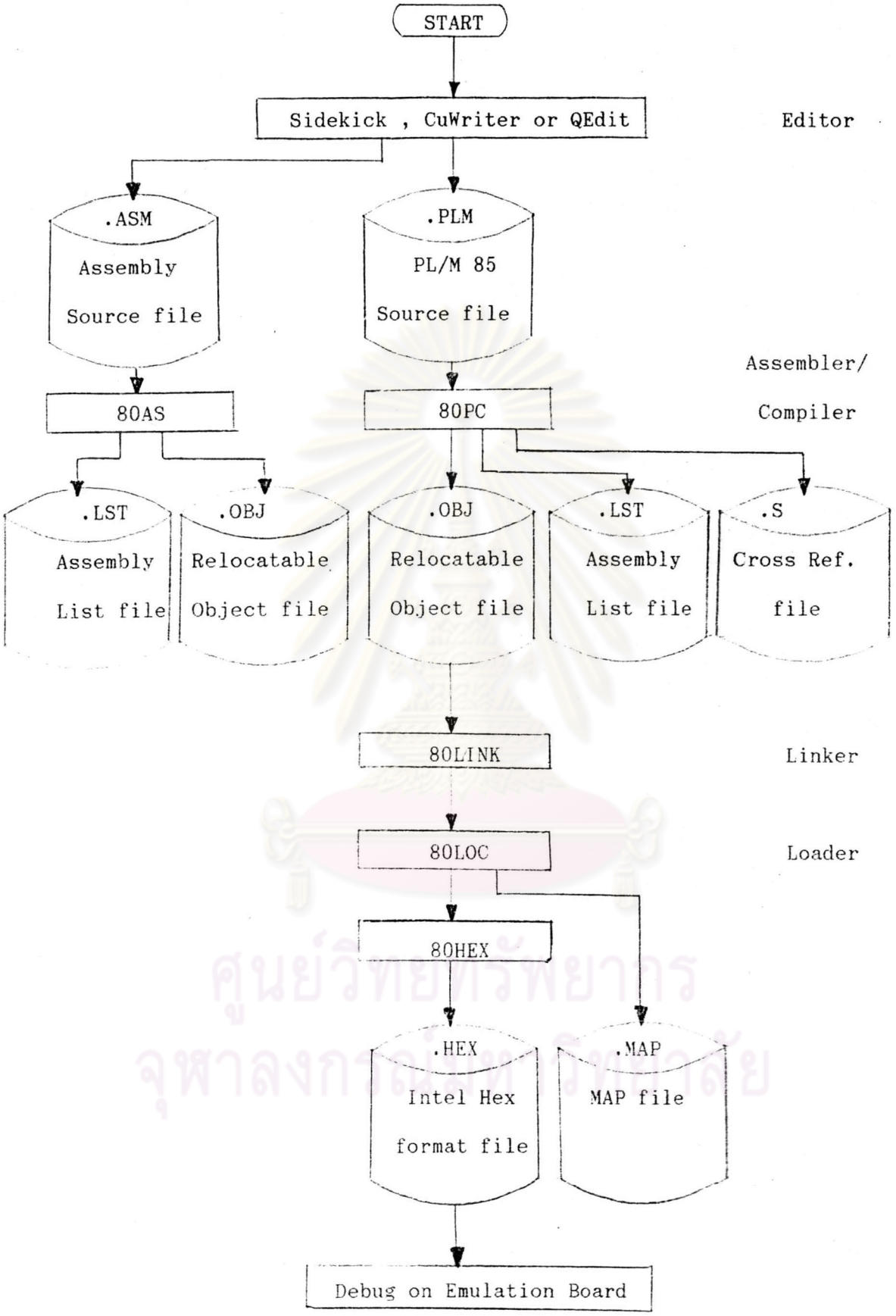


รูปที่ 7.4 แสดงขั้นตอนการออกแบบและพัฒนาซอฟต์แวร์



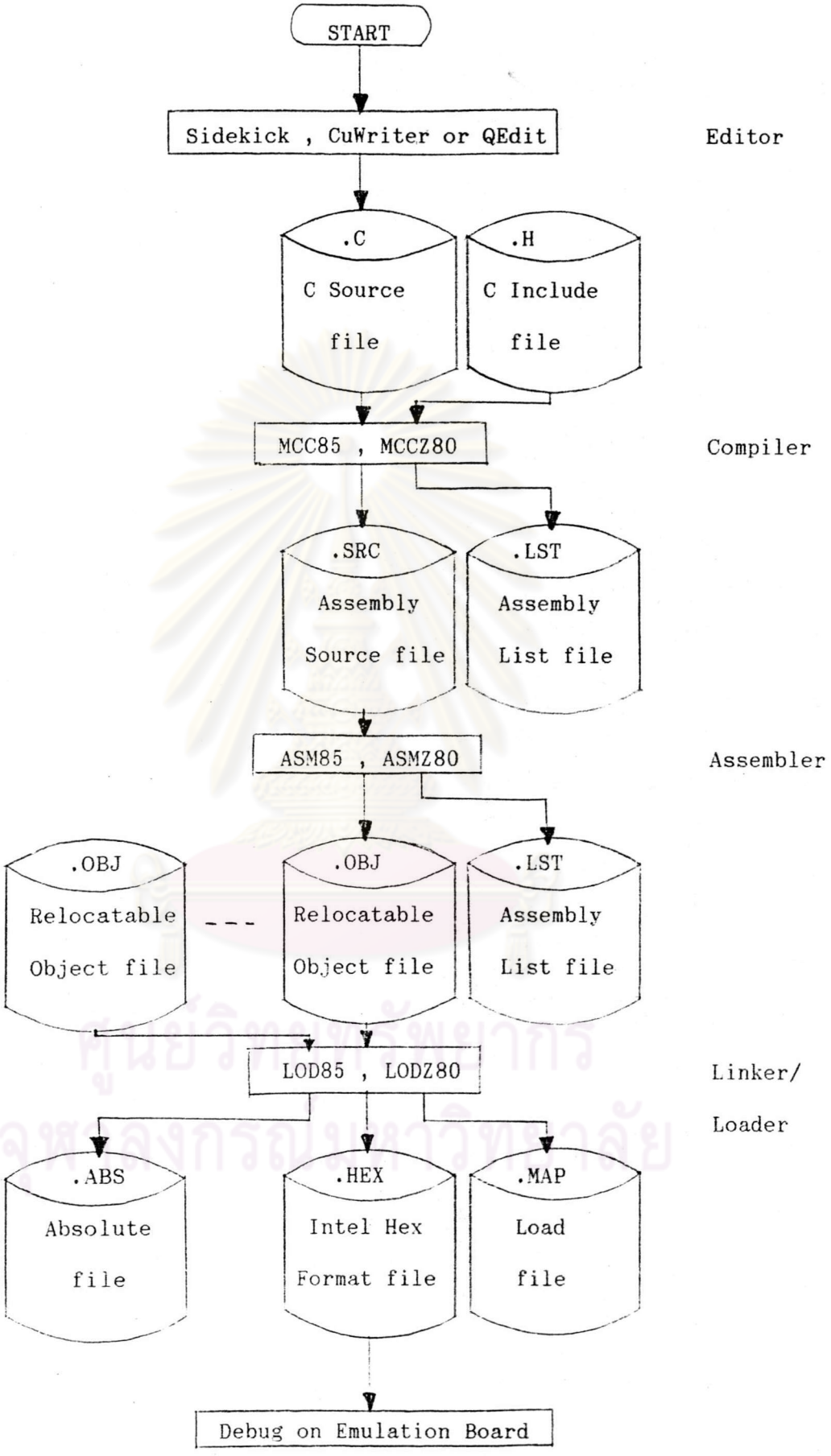
รูปที่ 7.5 แสดงขั้นตอนการเขียนโปรแกรมด้วยภาษาแอสเซมบลี

หากผู้ใช้เลือกเขียนโปรแกรมด้วยภาษา PL/M ซึ่งในที่นี้เป็น PL/M ของซีพียูเบอร์ 8085 ที่ใช้ชุดซอฟต์แวร์ของ 80/DS Expert-PL/M [22][23] จะต้องทำตามรูปที่ 7.6 คือ เริ่มจากการเขียนโปรแกรม Source file ผ่านเอดิเตอร์จะได้ไฟล์ที่มีนามสกุล .PLM จากนั้นเรียกใช้โปรแกรม 80PC เพื่อแปลโปรแกรมภาษา PL/M85 ให้เป็นโปรแกรมภาษาเครื่องจะได้ออกมา 3 ไฟล์คือ ไฟล์.OBJ ,ไฟล์ .LST ซึ่งเป็น Listing ในภาษาแอสเซมบลี และไฟล์.S ซึ่งเป็น Cross Reference file ที่บอกตำแหน่งของตัวแปรในหน่วยความจำ การเขียนโปรแกรมบางครั้งอาจมีความจำเป็นต้องเขียนโปรแกรมบางส่วนด้วยภาษาแอสเซมบลีและนำมาลิงก์กับโปรแกรมภาษา PL/M ฉะนั้นหลังจากเขียนโปรแกรมภาษาแอสเซมบลีผ่านทางเอดิเตอร์ได้เป็นไฟล์.ASM แล้ว ก็เรียกใช้โปรแกรม 80AS ในซอฟต์แวร์ชุดนี้แปลโปรแกรมให้เป็นภาษาเครื่องจะได้ไฟล์.OBJ และไฟล์.LST จากนั้นจึงเรียกใช้โปรแกรม 80LINK เพื่อลิงก์ไฟล์.OBJ และใช้โปรแกรม 80LOC กำหนดแอดเดรสที่แน่นอนให้โปรแกรม ซึ่งจะได้ไฟล์ .MAP ซึ่งบอกตำแหน่งของโปรแกรม ข้อมูล และแสต็กด้วย และได้ไฟล์ที่จะให้โปรแกรม 80HEX จัดรูปแบบเป็น Intel Hex format อีกที ดังนั้นจะได้ไฟล์.HEX ที่พร้อมจะนำไปโหลดลงบอร์ดอีมูเลชันเพื่อทำการดีบักต่อไป ในงานวิจัยครั้งนี้ได้ทำ Batch file ชื่อ PLM.BAT ไว้ให้เรียกใช้ไฟล์ต่าง ๆ ที่กล่าวมาแล้วได้ง่ายขึ้น โดยผู้ใช้เรียก PLM แล้วตามด้วยชื่อ Source file ก็จะได้ไฟล์ชื่อเดิมที่มีนามสกุล .HEX ออกมา



รูปที่ 7.6 แสดงขั้นตอนการเขียนโปรแกรมด้วยภาษา PL/M

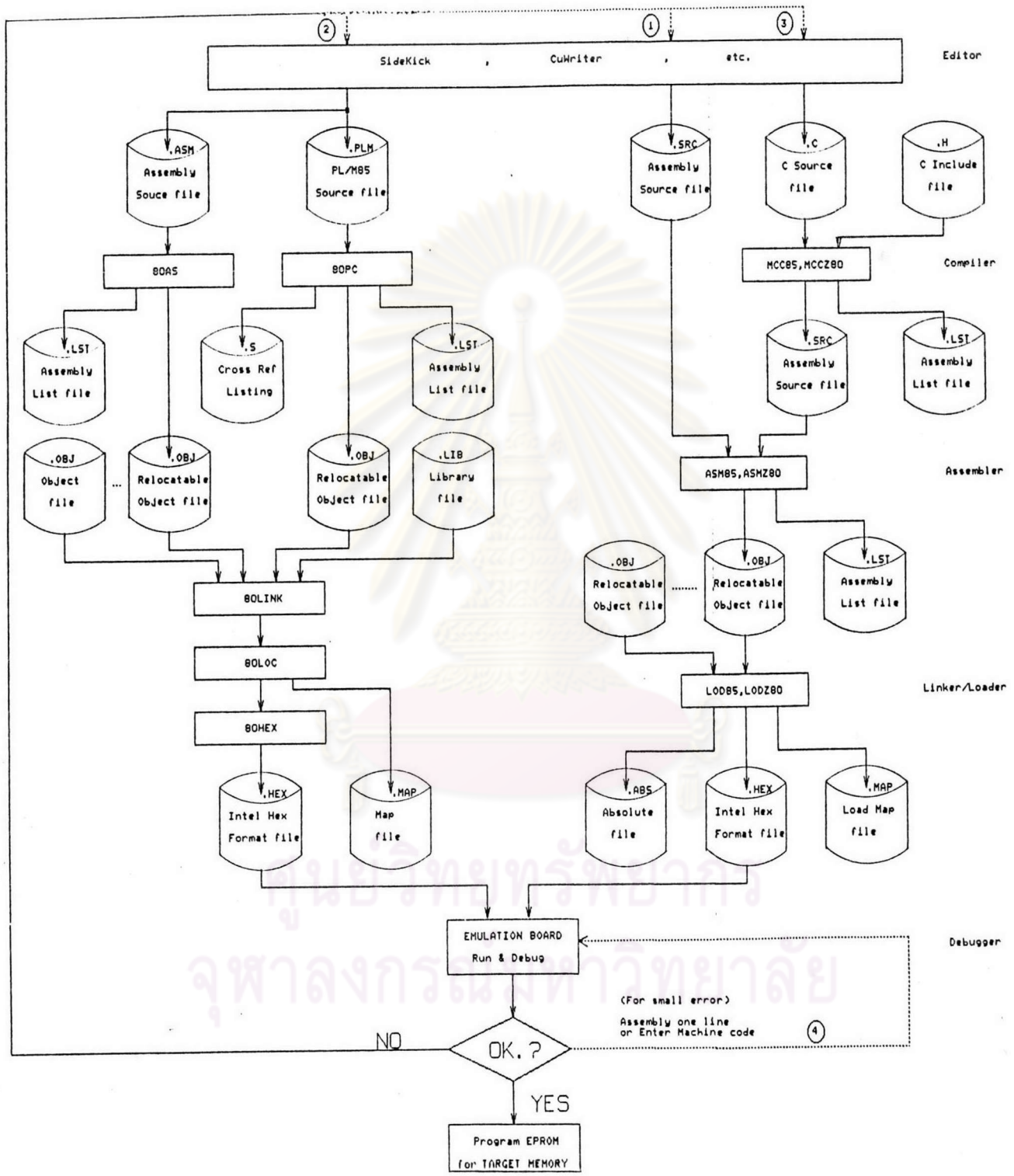
หากผู้ใช้เลือกเขียนโปรแกรมด้วยภาษา C จะต้องทำตามรูปที่ 7.7 ซึ่งในงานวิจัยครั้งนี้มีแต่ชุดซอฟต์แวร์ภาษา C ของซีพียูเบอร์ 8085 (MCC85 ของบริษัท Microtec Reserarch) ส่วนชุดซอฟต์แวร์ภาษา C ของซีพียูเบอร์ Z-80 (MCCZ80) นั้นแสดงไว้ให้ดูประกอบกัน การเขียนโปรแกรมเริ่มจากการเขียนโปรแกรมภาษา C เป็น Source file ผ่านทางเอดิเตอร์ได้เป็นไฟล์.C ซึ่งอาจมีการเรียกใช้ Include file ที่มีนามสกุล .H ด้วย จากนั้นก็มาแปลโดยใช้โปรแกรม MCC85 หรือ MCCZ80 จะได้ไฟล์ภาษาแอสเซมบลี .SRC กับไฟล์.LST แล้วจึงมาผ่านขบวนการเดียวกับการเขียนโปรแกรมด้วยภาษาแอสเซมบลีคือเรียกใช้ ASM85 หรือ ASMZ80 เพื่อแปลเป็นภาษาเครื่องและใช้ LOD85 หรือ LODZ80 กำหนดแอดเดรสของโปรแกรม ก็จะได้ไฟล์ที่เป็น Intel Hex format ออกมา พร้อมกับไฟล์ .MAP นั่นเอง



รูปที่ 7.7 แสดงขั้นตอนการเขียนโปรแกรมด้วยภาษา C

หลังจากที่ได้ไฟล์ในรูปแบบของ Intel Hex format แล้วก็นำมาทำการดีบักโดยใช้บอร์ดอีมูเลชันซึ่งได้กล่าวรายละเอียดการใช้งานไปแล้วในบทที่ 5 หากทดสอบโปรแกรมแล้วมีข้อผิดพลาด ผู้ใช้สามารถแก้ไขได้ 2 วิธี คือ ถ้าผิดพลาดน้อยอาจแก้ไขโดยการเปลี่ยนค่าในหน่วยความจำโดยตรงหรือเขียนโปรแกรมภาษาแอสเซมบลีที่ละบรรทัดผ่านทางแป้นพิมพ์เข้าไปแทนของเดิม แต่ถ้าผิดพลาดมากก็จะต้องกลับไปแก้ที่โปรแกรม Source file แล้วแปลงมาเป็น IntelHex format เพื่อทำการโหลดมาดีบักใหม่อีกครั้ง เมื่อดีบักโปรแกรมจนเป็นที่แน่ใจแล้วผู้ใช้สามารถเก็บโปรแกรมที่พัฒนาแล้วซึ่งอยู่ในแรมบนบอร์ดอีมูเลชันขึ้นเป็นไฟล์เพื่อจะนำไปโปรแกรมลงอีพรอม ขั้นตอนการโปรแกรมลงอีพรอมในระบบพัฒนาไมโครโปรเซสเซอร์ระดับบอร์ดอาศัยเครื่องโปรแกรมและซอฟต์แวร์ของบริษัท XelTex โดยเรียกใช้โปรแกรม HEXOBJ เพื่อแปลงไฟล์ที่ได้จากการพัฒนาเป็นโปรแกรม Binary file ก่อน จากนั้นจึงเรียกใช้โปรแกรม ROM212 และเปิดไฟเข้าเครื่องโปรแกรมอีพรอม เลือกชนิดและเบอร์ของอีพรอมให้ตรงกับที่จะโปรแกรม ตรวจสอบอีพรอมว่าว่างหรือไม่โดยใช้คำสั่ง Blank Check แล้วโหลดไฟล์ซึ่งอยู่ในรูปของไฟล์ .BIN มาไว้ในบัฟเฟอร์และสั่งโปรแกรมอีพรอม ฉะนั้นจะได้อีพรอมที่มีโปรแกรมที่พัฒนาแล้วอยู่ภายใน นำอีพรอมไปเสียบบนบอร์ดซีพียูและถอดบอร์ดอีมูเลชันออกเพื่อนำระบบไปใช้ในงานจริง และหากมีความจำเป็นต้องลดขนาดของระบบให้มีขนาดเล็กกะทัดรัดก็สามารถออกแบบระบบใหม่ โดยตัดฮาร์ดแวร์ส่วนที่ไม่จำเป็นขณะใช้งานออกไป

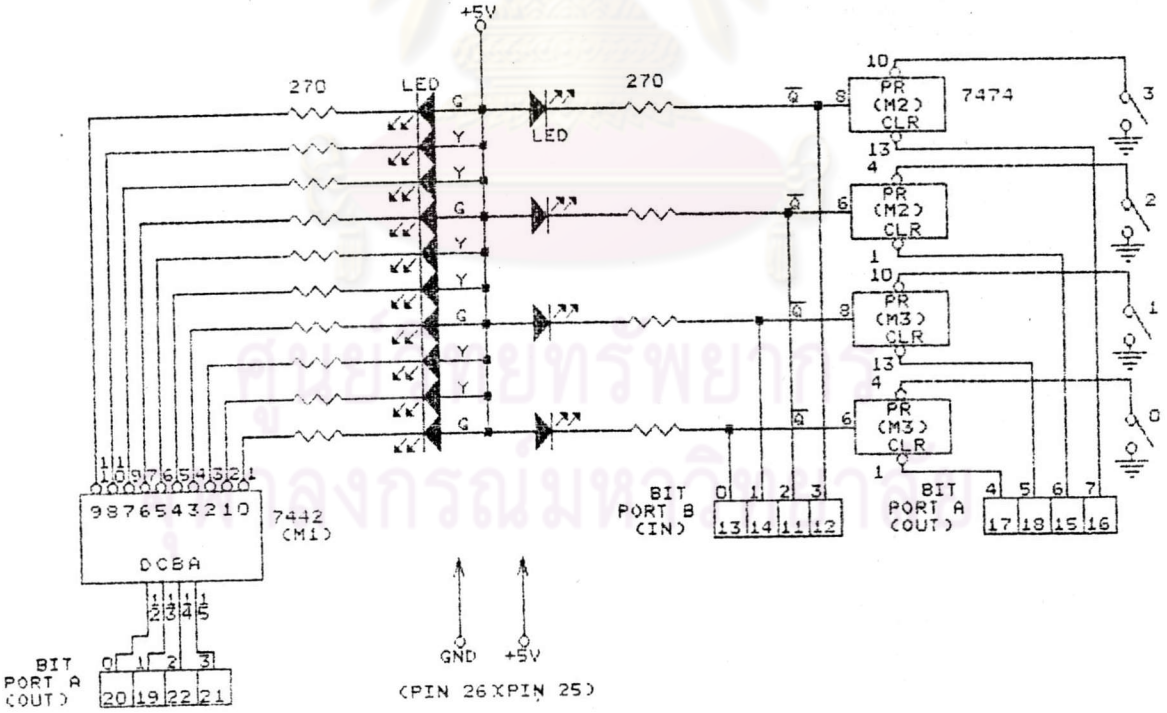
เราสามารถสรุปการเขียนโปรแกรมบนระบบพัฒนาไมโครโปรเซสเซอร์ระดับบอร์ดทั้งหมดได้ดังรูปที่ 7.8



รูปที่ 7.8 แสดงการเขียนโปรแกรมบนระบบพัฒนาไมโครโปรเซสเซอร์ระดับบอร์ดทั้งหมด

7.2 ตัวอย่างการประยุกต์ใช้งาน

ตัวอย่างการประยุกต์ใช้งานที่จะกล่าวต่อไป เป็นการนำเอาระบบพัฒนาไมโครโปรเซสเซอร์ระดับบอร์ดมาควบคุมระบบจำลองการทำงานของลิฟต์ 4 ชั้น ซึ่งมีปุ่มกดเรียกลิฟต์ในแต่ละชั้นและมี LED แสดงตำแหน่งของลิฟต์ เริ่มแรกจะออกแบบวงจรด้วยการนำเอาบอร์ดที่มีอยู่ในระบบมาประกอบเป็นวงจรร่วมกับชุดจำลองการทำงานของลิฟต์ [18] ซึ่งมีอยู่ในห้องปฏิบัติการ ในตัวอย่างครั้งนี้ใช้บอร์ดซีพียูและหน่วยความจำ Z-80 บอร์ดอีมูเลชัน และบอร์ดเชื่อมโยงไอ/โอแบบโปรแกรมได้โดยไม่จำเป็นต้องต่อวงจรเพิ่มเติมเลย เราออกแบบให้ใช้หน่วยความจำตั้งแต่แอดเดรส 0000H และเลือกใช้แรมบนบอร์ดอีมูเลชันก่อนโดยมีการต่อ Jumper ของแรมบนบอร์ดอีมูเลชันตามรูปที่ 4.6 และ 4.7 เลือกใช้พอร์ต 30H - 33H บนบอร์ดเชื่อมโยงไอ/โอแบบโปรแกรมได้ หลังจากประกอบวงจรเสร็จก็สามารถพัฒนาโปรแกรมบนระบบพัฒนาไมโครโปรเซสเซอร์ระดับบอร์ดร่วมกับเครื่องไมโครคอมพิวเตอร์ IBM PC ได้ วงจรของชุดจำลองการทำงานของลิฟต์แสดงได้ดังรูปที่ 7.9

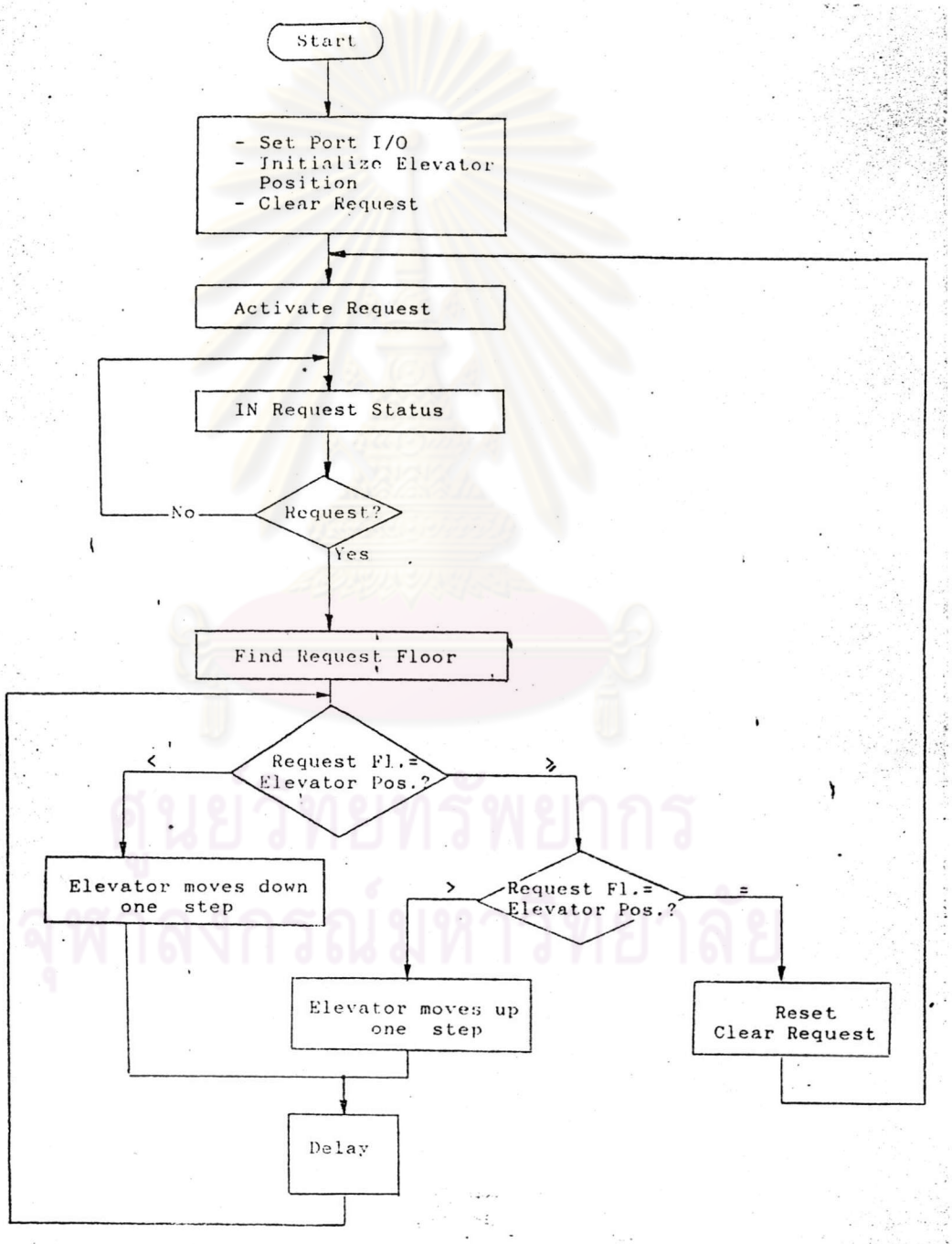


รูปที่ 7.9 แสดงวงจรของชุดจำลองการทำงานของลิฟต์

การเขียนโปรแกรมควบคุมลิฟต์จำลองนี้ เริ่มจากช่วงที่ 1 ของการออกแบบพัฒนาซอฟต์แวร์ คือ

1. ออกแบบขั้นตอนการทำงานของโปรแกรมเป็นแผนภูมิได้ดังรูปที่ 7.10

2. เลือกเขียนโปรแกรมด้วยภาษาแอสเซมบลี Z-80 และได้โปรแกรมควบคุมลิฟต์เขียนด้วยภาษาแอสเซมบลี Z-80 ดังรูปที่ 7.11



รูปที่ 7.10 แสดงขั้นตอนการทำงานของโปรแกรมควบคุมลิฟต์จำลอง

```
;Program Simulation of Elevator
;reg. B = Elevator Position
;     M = Request Floor
;8255 port address = 30h-33h
;
        ORG  0000H
START0: LD   SP,08800H      ;set stack pointer
        LD   A,082H         ;set port A : output port, B :input port
        OUT  (033H),A       ;set control port,clear request
        LD   B,000H         ;elevator at the ground floor at start
START1: LD   A,B
        OR   0F0H           ;activate request
        OUT  (030H),A
        LD   HL,FLOOR
START2: IN   A,(031H)       ;get the status of request keys
        OR   0F0H
        LD   C,A
        SUB  0FFH           ;have any request?
        JP   Z,START2       ;no loop back
START3: LD   A,C
        RRA                 ;determine the request floor
        LD   C,A
        JP   NC,DECIDE
        INC  HL
        JP   START3
DECIDE: LD   A,(HL)         ;get the request floor
        SUB  B
        JP   M,DOWN         ;elevator goes down
        JP   Z,RESET        ;no movement for the elevator
UP:     INC  B              ;elevator goes up
        LD   A,B
        OR   0F0H
        OUT  (030H).A       ;direction indicator LEDs turned on
        CALL DELAY
        CALL DELAY
        JP   DECIDE
DOWN:   DEC  B
        LD   A,B
        OR   0F0H
        OUT  (030H),A       ;direction indicator LEDs turned off
        CALL DELAY
        CALL DELAY
        JP   DECIDE
RESET:  LD   A,(HL)         ;no movement
        OUT  (030H),A
        JP   START1
DELAY:  PUSH BC             ;delay time loop
DELAY0: LD   B,0FFH
DELAY1: LD   C,0FFH
LOOP:   DEC  C
        JP   NZ,LOOP
        DEC  B
        JP   NZ,DELAY1
        POP  BC
        RET

FLOOR:  DB 00,03,06,09      ;data floor
        END
```

รูปที่ 7.11 แสดงโปรแกรมภาษาแอสเซมบลีของระบบลิฟต์จำลอง

3. เปิดเครื่องไมโครคอมพิวเตอร์ IBM PC แล้วเข้าสู่ไดเรกตอรี ASMZ80

4. เรียกโปรแกรม Q.EXE เพื่อป้อนโปรแกรมควบคุมสิพต์จำลองที่ได้ออกแบบไว้แล้วด้วยภาษาแอสเซมบลี Z-80 เข้าสู่ไฟล์ชื่อ LIFT0.ASM

5. ทำการแปลโปรแกรมโดยใช้ ASMZ80 ตามรูปที่ 7.12 จะได้โปรแกรม LIFT0.OBJ และ LIFT0.LST ดังรูปที่ 7.13

```
C:\STD-MDS\ASMZ80>ASMZ80
Paragon ASMZ80 Assembler - Version 4.3D
Copyright (c) 1983,1984 Microtec Research Inc.
ALL RIGHTS RESERVED.    Serial Number 3-003094

Source filename              [.SRC]:LIFT0.ASM
Object filename          [C:LIFT0.OBJ]:LIFT0
List filename              [C:NUL.LST]:LIFT0

  ASSEMBLER ERRORS =     0
C:\STD-MDS\ASMZ80>LODZ80
Paragon LODZ80 Loader - Version 4.3D
Copyright (c) 1983,1984 Microtec Research Inc.
ALL RIGHTS RESERVED.    Serial Number 3-003094

Object/Command File          [.OBJ]:LIFT0
Output Object File       [C:LIFT0.ABS]:LIFT0.HEX
Map Filename               [C:NUL.MAP]:LIFT0
```

รูปที่ 7.12 แสดงการแปลโปรแกรม LIFT0.ASM โดยใช้ ASMZ80 และ LODZ80

```
MICROTEC ASMZ80 V4.3D                    PAGE  1

ERR  LINE  ADDR  OBJ

       1                        ;Program Simulation of Elevator
       2                        ;reg. B = Elevator Position
       3                        ;     M = Request Floor
       4                        ;8255 port address = 30h-33h
       5                        ;
       6                               ORG  0000H
       7  0000  31 00 88        START0: LD   SP,08800H      ;set stack pointer
       8  0003  3E 82                   LD   A,082H         ;set port A : output port, B :input port
       9  0005  D3 33                   OUT  (033H),A       ;set control port,clear request
      10  0007  06 00                   LD   B,000H         ;elevator at the ground floor at start
      11  0009  78              START1: LD   A,B
      12  000A  F6 F0                   OR   0F0H           ;activate request
      13  000C  D3 30                   OUT  (030H),A
      14  000E  21 60 00                LD   HL,FLOOR
      15  0011  DB 31             START2: IN   A,(031H)       ;get the status of request keys
      16  0013  F6 F0                   OR   0F0H
      17  0015  4F                      LD   C,A
      18  0016  D6 FF                   SUB  0FFH           ;have any request?
      19  0018  CA 11 00                JP   Z,START2       ;no loop back
      20  001B  79              START3: LD   A,C
      21  001C  1F                      RRA                 ;determine the request floor
      22  001D  4F                      LD   C,A
      23  001E  D2 25 00                JP   NC,DECIDE
      24  0021  23                      INC  HL
      25  0022  C3 1B 00                JP   START3
      26  0025  7E              DECIDE: LD   A,(HL)         ;get the request floor
      27  0026  90                      SUB  B
      28  0027  FA 3C 00                JP   M,DOWN         ;elevator goes down
      29  002A  CA 4B 00                JP   Z,RESET        ;no movement for the elevator
      30  002D  04              UP:     INC  B              ;elevator goes up
      31  002E  78                      LD   A,B
      32  002F  F6 F0                   OR   0F0H
      33  0031  D3 30                   OUT  (030H),A       ;direction indicator LEDs turned on
      34  0033  CD 51 00                CALL DELAY
      35  0036  CD 51 00                CALL DELAY
      36  0039  C3 25 00                JP   DECIDE
      37  003C  05              DOWN:   DEC  B
      38  003D  78                      LD   A,B
      39  003E  F6 F0                   OR   0F0H
      40  0040  D3 30                   OUT  (030H),A       ;direction indicator LEDs turned off
      41  0042  CD 51 00                CALL DELAY
      42  0045  CD 51 00                CALL DELAY
      43  0048  C3 25 00                JP   DECIDE
      44  004B  7E              RESET:  LD   A,(HL)         ;no movement
      45  004C  D3 30                   OUT  (030H),A
      46  004E  C3 09 00                JP   START1
      47  0051  C5              DELAY:  PUSH BC             ;delay time loop
      48  0052  06 FF           DELAY0: LD   B,0FFH
      49  0054  0E FF           DELAY1: LD   C,0FFH
      50  0056  0D              LOOP:   DEC  C
```

รูปที่ 7.13 แสดงโปรแกรม LIFT0.LST

```
          MICROTEC ASMZ80 V4.3D                          PAGE   2

ERR  LINE  ADDR  OBJ

      51  0057  C2 56 00              JP   NZ,LOOP
      52  005A  05                    DEC  B
      53  005B  C2 54 00              JP   NZ,DELAY1
      54  005E  C1                    POP  BC
      55  005F  C9                    RET
      56  0060
      57  0060  00 03 06 09     FLOOR: DB 00,03,06,09     ;data floor
      58  0064                        END

  ASSEMBLER ERRORS =    0

          MICROTEC ASMZ80 V4.3D                          PAGE   3


                              SYMBOL TABLE
DECIDE          0025     DELAY          0051     DELAY0         0052
DELAY1          0054     DOWN           003C     FLOOR          0060
LOOP            0056     MEMORY       M 0000     NARG           0000
RESET           004B     STACK        S 0000     START0         0000
START1          0009     START2         0011     START3         001B
UP              002D
```

รูปที่ 7.13 แสดงโปรแกรม LIFT0.LST (ต่อ)

6. ทำการกำหนดแอดเดรสที่แน่นอนให้กับโปรแกรม LIFT0.OBJ โดยใช้โปรแกรม LODZ80 ตามรูปที่ 7.12 จะได้ไฟล์ LIFT0.HEX ในรูปที่ 7.14 และ LIFT0.MAP ในรูปที่ 7.15

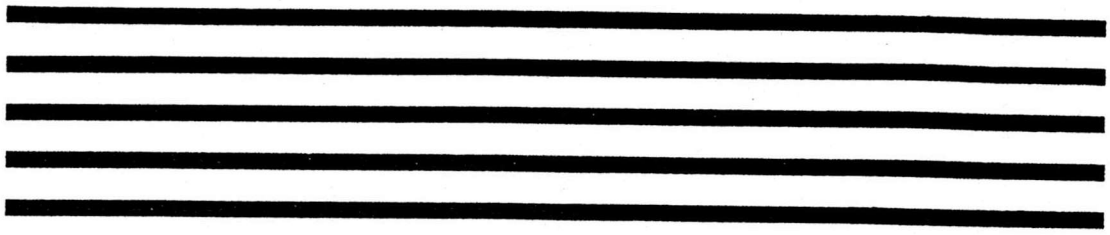

```
:10000000310088E82D333060078F6F0D330216089
:1000100000DB31F6F04FD6FFCA1100791F4FD22511
:10002000023C31B007E90FA3C00CA4B000478F604
:10003000F0D330CD5100CD5100C325000578F6F046
:10004000D330CD5100CD5100C325007ED330C3093C
:1000500000C506FF03FF0DC2560005C25400C1C9FF
:04006000000306098A
:00000001FF
```

รูปที่ 7.14 แสดงโปรแกรม LIFT0.HEX

```
**LOAD MAP**

  MODULE              CODE  DATA  EXTRA

  MODULE              0000  0000  0000
  //                  0000  0000  0000
  STACK               0000
  //                  0000
  MEMORY              0000
  //                  0000


  ABSOLUTE SEGMENTS

                      0000  0063

 **PUBLIC SYMBOLS TABLE

STKSZ         0000

 **ENTRY POINT    0000H

 **LOAD COMPLETED
```

รูปที่ 7.15 แสดงโปรแกรม LIFT0.MAP

เมื่อได้โปรแกรม LIFT0.HEX ซึ่งเป็นโปรแกรมในรูปแบบ Intel Hex format ก็จะเข้าสู่ช่วงที่ 2 ของการพัฒนาซอฟต์แวร์ คือ ใช้บอร์ดอีมูเลชันในการดีบักโปรแกรมร่วมกับโปรแกรม PROCOMM+ บนเครื่องไมโครคอมพิวเตอร์ IBM PC โดยต่อสายเคเบิล RS-232C จากบอร์ดอีมูเลชันเข้าสู่พอร์ตอนุกรม COM1 หรือ COM2 ของเครื่องไมโครคอมพิวเตอร์ ในที่นี้ใช้พอร์ต COM2 จากนั้นเข้าสู่ไดเรกตอรี PROCOMM แล้วเรียกใช้โปรแกรม PCPLUS.EXE จะได้รูปที่ 7.16 ปรากฏบนหน้าจอภาพ



PROCOMM PLUS (R)

Intuitive Communications (tm)

» PROCOMM PLUS - Version 1.1B «
Copyright (C) 1987, 1988 DATASTORM TECHNOLOGIES, INC.
All Rights Reserved
UNAUTHORIZED DUPLICATION PROHIBITED

PRESS ANY KEY TO ENTER TERMINAL MODE

รูปที่ 7.16 แสดงจอภาพเมื่อเรียกใช้ PCPLUS

ต่อไปตั้งค่าการติดต่อสื่อสารระหว่างโปรแกรม PROCOMM+ กับบอร์ดอีมูเลชันซึ่งจะตั้งค่าเพียงครั้งเดียวก่อนทำการดีบักโปรแกรม โดยมีขั้นตอนดังนี้

1. ตั้งค่าการติดต่อกับพอร์ตอนุกรม COM1 หรือ COM2 โดยการกด Alt-P จะปรากฏหน้าจอตามรูปที่ 7.17

2. เลือกกดหมายเลข 2 - 5 เพื่อกำหนดอัตราเร็วในการรับส่งข้อมูล ในที่นี้ใช้ 9,600 บิตต่อวินาที จึงกดหมายเลข 2

3. กด Alt-N เพื่อกำหนดรูปแบบการรับส่งข้อมูลเป็นแบบ

No Parity , 8 bit/char และ 1 Stop bit

4. เลือกกด F1 หรือ F2 ตามการต่อสาย RS-232C กับบอร์ดอีมูเลชัน ในที่นี้กดปุ่ม F2 เพราะใช้พอร์ต COM2

5. กด Alt-S เพื่อเก็บค่า LINE/PORT SETUP ที่ตั้งไว้แล้วจากข้อ 2 - ข้อ 4

PROCOMM PLUS Ready!

```
                    CURRENT SETTINGS:  9600,N,8,1,COM2

BAUD RATE     PARITY        DATA BITS     STOP BITS     PORT

1)     300    N) NONE       Alt-7) 7      Alt-1) 1      F1) COM1
2)    1200    E) EVEN       Alt-8) 8      Alt-2) 2      F2) COM2
3)    2400    O) ODD                                    F3) COM3
4)    4800    M) MARK                                   F4) COM4
5)    9600    S) SPACE                                  F5) COM5
6)   19200                                              F6) COM6
7)   38400                                              F7) COM7
8)   57600    Alt-N) N/8/1                              F8) COM8
9)  115200    Alt-E) E/7/1

Esc) Exit     Alt-S) Save and Exit   YOUR CHOICE:
```

LINE/PORT SETUP

รูปที่ 7.17 แสดงการตั้งค่าการติดต่อพอร์ตอนุกรม COM2

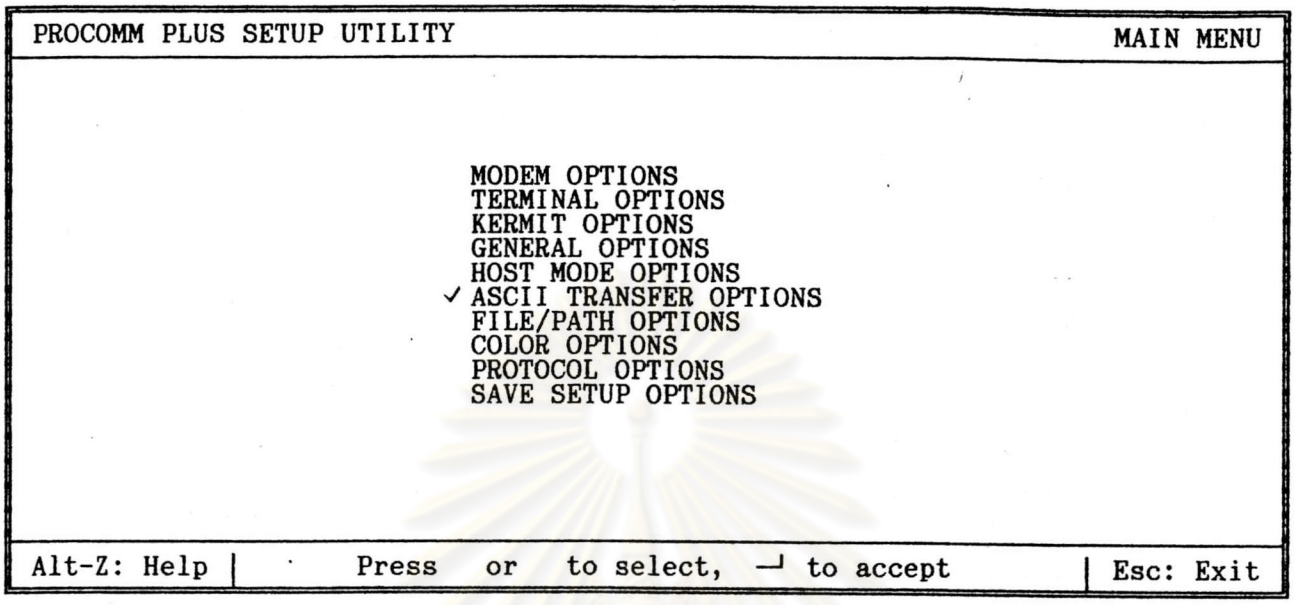
6. ตั้งค่าเกี่ยวกับการรับส่งตัวอักษรกับบอร์ดอีมูเลชัน โดยกด Alt-S จะปรากฏ Main Menu ดังรูปที่ 7.18 จากนั้นเลือกเข้าสู่ ASCII TRANSFER OPTIONS เพื่อกำหนดค่าตามรูปที่ 7.19

7. กด D เพื่อเลือกระยะเวลารอการรับส่งตัวอักษรแต่ละครั้ง ในที่นี้ใช้ค่า 0

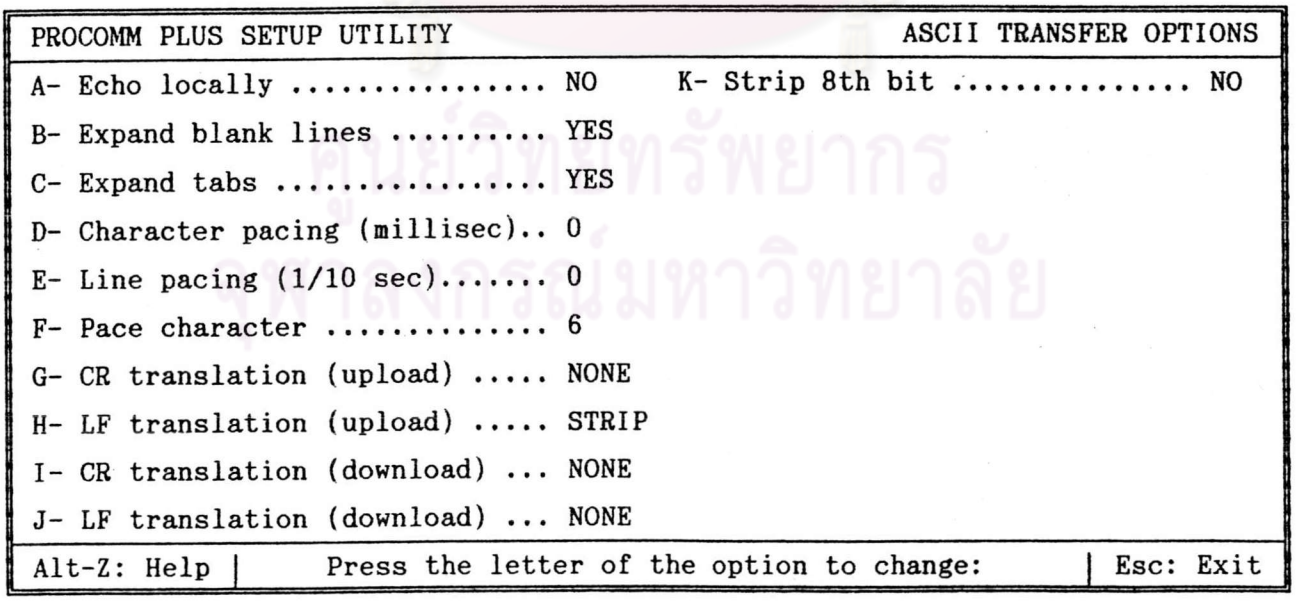
8. กด E เพื่อเลือกระยะเวลารอการขึ้นบรรทัดใหม่ ในที่นี้ใช้ค่า 0 เช่นกัน

9. กด F เพื่อเลือกตัวอักษรที่ให้ PROCOMM รอตัวอักษรนี้ก่อนจึงขึ้นบรรทัดใหม่ได้ ในที่นี้ให้รอ ASCII ตัวที่ 6 คือ ACK จากบอร์ดอีมูเลชัน จึงใส่ค่า 6 แล้วกด Enter

10. กด ESC ให้ออกไปที่ Main Menu แล้วเลือก SAVE SETUP OPTIONS เพื่อเก็บค่าที่กำหนดจากข้อ 7 - ข้อ 9 ไว้สำหรับ ASCII TRANSFER OPTIONS ซึ่งจะต้องตั้งค่าในครั้งแรกที่ใช้โปรแกรม PROCOMM

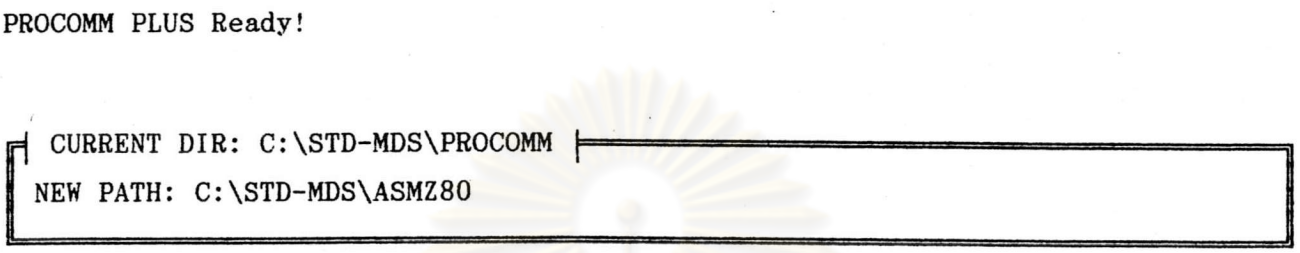


รูปที่ 7.18 แสดง MAIN MENU ของโปรแกรม PROCOMM



รูปที่ 7.19 แสดงการกำหนดการรับส่งตัวอักษรกับบอร์ดอีมูเลชัน

11. เปลี่ยนไดเรกตอรีข้อมูลให้ไปเรียกโปรแกรมที่ไดเรกตอรี ซึ่งมีโปรแกรมที่ต้องการจะพัฒนาอยู่ ในตัวอย่างนี้โปรแกรมที่ต้องการจะพัฒนาอยู่ในไดเรกตอรี ASMZ80 จึงกำหนดดังรูปที่ 7.20 โดยกด Alt-F7 และ C:\STD-MDS\ASMZ80



รูปที่ 7.20 แสดงจอภาพการเปลี่ยนไดเรกตอรีข้อมูลเมื่อกด Alt-F7

12. เมื่อตั้งค่าการติดต่อต่าง ๆ เรียบร้อยแล้ว ก็เปิดไฟเข้าระบบพัฒนาไมโครโปรเซสเซอร์ระดับบอร์ด หากการติดต่อสำเร็จจะปรากฏจอภาพดังรูปที่ 7.21

13. ทำการรีเซตระบบด้วยคำสั่ง X และใส่ค่าในหน่วยความจำบริเวณที่ต้องการโหลดโปรแกรมมาทดสอบ ในที่นี้ใช้คำสั่ง F ใส่ค่า 0 ตั้งแต่แอดเดรส 0000H - 00FFH

14. โหลดโปรแกรม LIFTO.HEX ลงบนบอร์ดอีมูเลชัน โดยการกด L และ <CR> จากนั้นกดปุ่ม Page Up จะได้จอภาพดังรูปที่ 7.22

15. กดหมายเลข 4 เพื่อเลือกการส่งข้อมูลแบบ ASCII แล้วกด <CR> จะได้จอภาพดังรูปที่ 7.23 ให้ใส่ชื่อไฟล์ LIFTO.HEX

16. เมื่อโหลดเรียบร้อยจะปรากฏดังรูปที่ 7.24

```
PROCOMM PLUS Ready!

STD Debugger for Z80
Assemble    A [address]
Breakpoint  B [address]
Clearbreak  C
Dump        D [address¦range]
Enter       E address [list]
Fill        F range list
Go          G [address]
Help        H
Input       I address
Load        L
Output      O address byte
Register    R [registername]
Trace       T
Unassemble  U [S][address¦range]
Write       W range
reset       X
>


 Alt-Z FOR HELP| ANSI     | FDX |  9600 N81 | LOG CLOSED | PRINT OFF | OFF-LINE
```

รูปที่ 7.21 แสดงจอภาพเมื่อเปิดไฟเข้าระบบพัฒนาไมโครโปรเซสเซอร์ระดับบอร์ด

```
PROCOMM PLUS Ready!

ST┌──┤ Upload Protocols ├──────────────────────────────────────────────┐
As│                                                                    │
Br│  1) XMODEM        5) TELINK        9) WXMODEM       13) YMODEM-G BATCH
Cl│  2) KERMIT        6) MODEM7       10) IMODEM        14) EXTERN 1
Du│  3) YMODEM        7) SEALINK      11) YMODEM-G      15) EXTERN 2
En│  4) ASCII         8) COMPUSERVE B 12) YMODEM BATCH  16) EXTERN 3
Fi│                                                                    │
Go│  Your Selection: 4   (or press ENTER for XMODEM)                   │
He└────────────────────────────────────────────────────────────────────┘
Input       I address
Load        L
Output      O address byte
Register    R [registername]
Trace       T
Unassemble  U [S][address¦range]
Write       W range
reset       X
>X
>F 00 FF 0
>L
Use load file to COM utility of your terminal.
 Alt-Z FOR HELP| ANSI     | FDX |  9600 N81 | LOG CLOSED | PRINT OFF | OFF-LINE
```

รูปที่ 7.22 แสดงจอภาพเมื่อใช้คำสั่ง L แล้วกดปุ่ม Page UP

```
PROCOMM PLUS Ready!

STD Debugger for Z80
Assemble    A [address]
Breakpoint  B [address]
Clearbreak  C
Dump        D [address¦range]
Enter       E address [list]
Fill        F range list
Go          G [address]
Help        H
Input       I address
Load        L
Outpu       O address byte
Regis       R [registername]
Trace       T
Unassemble  U [S][address¦range]
Write       W range
reset       X
>X
>F 00 FF 0
>L
Use load file to COM utility of your terminal.
 Alt-Z FOR HELP| ANSI     | FDX |  9600 N81 | LOG CLOSED | PRINT OFF | OFF-LINE
```


รูปที่ 7.23 แสดงจอภาพเมื่อใช้คำสั่ง L และใส่ชื่อไฟล์ LIFT0.HEX

```
Assemble    A [address]
Breakpoint  B [address]
Clearbreak  C
Dump        D [address¦range]
Enter       E address [list]
Fill        F range list
Go          G [address]
Help        H
Input       I address
Load        L
Output      O address byte
Register    R [registername]
Trace       T
Unassemble  U [S][address¦range]
Write       W range
reset       X
>X
>F 00 FF 0
>L
Use load file to COM utility of your terminal.

Load ready.
>
>
 Alt-Z FOR HELP| ANSI     | FDX |  9600 N81 | LOG CLOSED | PRINT OFF | OFF-LINE
```

รูปที่ 7.24 แสดงจอภาพเมื่อโหลดไฟล์ LIFT0.HEX เรียบร้อยแล้ว

17. ทำการตรวจสอบว่าโหลดโปรแกรมได้ถูกต้องด้วยคำสั่ง U ดังรูปที่ 7.25

```
>U 0 27
0000 310088    LD   SP,8800
0003 3E82      LD   A,82
0005 D333      OUT  (33),A
0007 0600      LD   B,00
0009 78        LD   A,B
000A F6F0      OR   F0
000C D330      OUT  (30),A
000E 216000    LD   HL,0060
0011 DB31      IN   A,(31)
0013 F6F0      OR   F0
0015 4F        LD   C,A
0016 D6FF      SUB  FF
0018 CA1100    JP   Z,0011
001B 79        LD   A,C
001C 1F        RRA
001D 4F        LD   C,A
001E D22500    JP   NC,0025
0021 23        INC  HL
0022 C31B00    JP   001B
0025 7E        LD   A,(HL)
0026 90        SUB  B
0027 FA3C00    JP   M,003C
>
 Alt-Z FOR HELP| ANSI      | FDX |  9600 N81 | LOG CLOSED | PRINT OFF | OFF-LINE
```

รูปที่ 7.25 แสดงการติสแอสเซมเบลอร์โปรแกรม LIFT0.HEX บนบอร์ดอีมูเลชัน

18. ทำการดีบักต่าง ๆ ตามรูปที่ 7.26 และ 7.27 เช่น ดูค่ารีจิสเตอร์ ทดลองตั้งจุดหยุดและทดลองรันโปรแกรมดู

```
>R
  PC   A  F(SZHPNC) B  C  D  E  H  L I IFF SP   IX   IY   AF'  BC'  DE'  HL'
 0000 00 00 000000 00 00 00 00 00 00 00 0 0000 0000 0000 0000 0000 0000 0000
>R A
A   00
55
>R
  PC   A  F(SZHPNC) B  C  D  E  H  L I IFF SP   IX   IY   AF'  BC'  DE'  HL'
 0000 55 00 000000 00 00 00 00 00 00 00 0 0000 0000 0000 0000 0000 0000 0000
>T
  PC   A  F(SZHPNC) B  C  D  E  H  L I IFF SP   IX   IY   AF'  BC'  DE'  HL'
 0003 55 00 000000 00 00 00 00 00 00 00 0 8800 0000 0000 0000 0000 0000 0000
>T
  PC   A  F(SZHPNC) B  C  D  E  H  L I IFF SP   IX   IY   AF'  BC'  DE'  HL'
 0005 82 00 000000 00 00 00 00 00 00 00 0 8800 0000 0000 0000 0000 0000 0000
>T
  PC   A  F(SZHPNC) B  C  D  E  H  L I IFF SP   IX   IY   AF'  BC'  DE'  HL'
 0007 82 00 000000 00 00 00 00 00 00 00 0 8800 0000 0000 0000 0000 0000 0000
>T
  PC   A  F(SZHPNC) B  C  D  E  H  L I IFF SP   IX   IY   AF'  BC'  DE'  HL'
 0009 82 00 000000 00 00 00 00 00 00 00 0 8800 0000 0000 0000 0000 0000 0000
>B
Break Point Clear
>
 Alt-Z FOR HELP| ANSI      | FDX |  9600 N81 | LOG CLOSED | PRINT OFF | OFF-LINE
```

รูปที่ 7.26 แสดงตัวอย่างการดีบักโปรแกรมด้วยคำสั่ง R ,T ,B

```
>G
  PC   A  F(SZHPNC) B  C  D  E  H  L I IFF SP   IX   IY   AF'  BC'  DE'  HL'
 0056 F4 3E 001110 46 7F 00 00 00 62 00 0 87FC 0000 0000 0000 0000 0000 0000
>B 11
>B
Break Point Address 0011
>G
  PC   A  F(SZHPNC) B  C  D  E  H  L I IFF SP   IX   IY   AF'  BC'  DE'  HL'
 0013 02 A4 100100 06 FE 00 00 00 60 00 0 8800 0000 0000 0000 0000 0000 0000
>B
Break Point Address 0011
>C
>B
Break Point Clear
>R PC
PC  0013
0
>R
  PC   A  F(SZHPNC) B  C  D  E  H  L I IFF SP   IX   IY   AF'  BC'  DE'  HL'
 0000 02 A4 100100 06 FE 00 00 00 60 00 0 8800 0000 0000 0000 0000 0000 0000
>W 0 63
Use write COM to file utility of your terminal.
Strike a key when ready..

 Alt-Z FOR HELP| ANSI      | FDX |  9600 N81 | LOG CLOSED | PRINT OFF | OFF-LINE
```

รูปที่ 7.27 แสดงตัวอย่างการเก็บโปรแกรมที่ดีบักเสร็จแล้วด้วยคำสั่ง W

19. หลังจากดีบักโปรแกรมเรียบร้อยแล้ว ทำการเก็บโปรแกรมจากหน่วยความจำลงสู่ดิสค์ในไฟล์ LIFT.HEX โดยใช้คำสั่ง W ตามด้วยแอดเดรสของโปรแกรมดังรูปที่ 7.27 จากนั้นกดปุ่ม Page Down ได้จอภาพดังรูปที่ 7.28 เลือกส่งแบบ ASCII และใส่ชื่อไฟล์จะได้จอภาพดังรูปที่ 7.29 เมื่อเก็บโปรแกรมลงสู่ไฟล์เรียบร้อยแล้ว ให้กดปุ่ม ESC อีกครั้งหนึ่ง

```
>G
  PC   A  F(SZHPNC) B  C  D  E  H  L I IFF SP   IX   IY   AF'  BC'  DE'  HL'
 0┌──┤ Download Protocols - 1554432 bytes free ├──────────────────────────────┐
>│                                                                            │
>│   1) XMODEM        5) TELINK        9) WXMODEM       13) YMODEM-G BATCH    │
Br│   2) KERMIT       6) MODEM7       10) IMODEM        14) EXTERN 1          │
>│   3) YMODEM       7) SEALINK      11) YMODEM-G      15) EXTERN 2          │
 │   4) ASCII        8) COMPUSERVE B 12) YMODEM BATCH  16) EXTERN 3          │
 0                                                                            │
>│  Your Selection: 4   (or press ENTER for ASCII)                            │
Br└────────────────────────────────────────────────────────────────────────────┘
>C
>B
Break Point Clear
>R PC
PC  0013
0
>R
  PC   A  F(SZHPNC) B  C  D  E  H  L I IFF SP   IX   IY   AF'  BC'  DE'  HL'
 0000 02 A4 100100 06 FE 00 00 00 60 00 0 8800 0000 0000 0000 0000 0000 0000
>W 0 63
Use write COM to file utility of your terminal.
Strike a key when ready..

 Alt-Z FOR HELP| ANSI      | FDX |  9600 N81 | LOG CLOSED | PRINT OFF | OFF-LINE
```

รูปที่ 7.28 แสดงจอภาพเมื่อกดปุ่ม Page Down

```
Break Point Address 0011
>C
>B
Break Point Clear
>R PC
PC  0013
0
>R
  PC   A  F(SZHPNC) B  C  D  E  H  L I IFF SP   IX   IY   AF'  BC'  DE'  HL'
 0000 02 A4 100100 06 FE 00 00 00 60 00 0 8800 0000 0000 0000 0000 0000 0000
>W 0 63
Use write COM to file utility of your terminal.
Strike a key when ready..
:100000003100883E82D333060078F6F0D330216089
:1000100000DB31F6F04FD6FFCA1100791F4FD22511
:100020000023C31B007E90FA3C00CA4B000478F604
:10003000F0D330CD5100CD5100C325000578F6F046
:10004000D330CD5100CD5100C325007ED330C3093C
:1000500000C506FF0EFF0DC2560005C25400C1C9FF
:04006000000306098A
:00000001FF

Write ready.
>
 LINE: 10           ASCII FILE DOWNLOAD  -  PRESS ESC TO END
```

รูปที่ 7.29 แสดงจอภาพเมื่อเก็บโปรแกรมลงสู่ไฟล์เรียบร้อยแล้ว

```
Break Point Address 0011
>C
>B
Break┌ EXIT TO DOS? (Y/N) Yes ┐
>R P└                         ┘
PC  0
0
>R
  PC   A  F(SZHPNC) B  C  D  E  H  L I IFF SP   IX   IY   AF'  BC'  DE'  HL'
 0000 02 A4 100100 06 FE 00 00 00 60 00 0 8800 0000 0000 0000 0000 0000 0000
>W 0 63
Use write COM to file utility of your terminal.
Strike a key when ready..
:100000003100883E82D333060078F6F0D330216089
:1000100000DB31F6F04FD6FFCA1100791F4FD22511
:100020000023C31B007E90FA3C00CA4B000478F604
:10003000F0D330CD5100CD5100C325000578F6F046
:10004000D330CD5100CD5100C325007ED330C3093C
:1000500000C506FF0EFF0DC2560005C25400C1C9FF
:04006000000306098A
:00000001FF

Write ready.
>
 Alt-Z FOR HELP| ANSI      | FDX |  9600 N81 | LOG CLOSED | PRINT OFF | OFF-LINE
```

รูปที่ 7.30 แสดงจอภาพเมื่อต้องการออกจากโปรแกรม PROCOMM+

20. กดปุ่ม ESC เมื่อต้องการออกจากโปรแกรม PROCOMM+ จะได้จอภาพในรูปที่ 7.30 หากต้องการออกจริงให้กด Y

เมื่อได้โปรแกรมที่พัฒนาจนเป็นที่แน่ใจแล้ว ก็เข้าสู่ช่วงที่ 3 ของการออกแบบพัฒนาซอฟต์แวร์ คือ นำโปรแกรมมาลงอีพรอมโดยใช้เครื่องโปรแกรม UNIPRO ของบริษัท XelTex ซึ่งก่อนอื่นต้องนำโปรแกรมที่ได้มาแปลงเป็นโปรแกรมในรูปของไบนารีไฟล์ก่อนโดยใช้โปรแกรม HEXOBJ ตามรูปที่ 7.31 จะได้ไฟล์ออกมาชื่อ LIFT0.BIN

```
C:\STD-MDS\UNIPRO>HEXOBJ

   HI-LO  HEX TO OBJECT CONVERTER V3.1   1988
-----------------------------------------------

HEX file name:..\ASMZ80\LIFT0.HEX
OBJ file name:..\ASMZ80\LIFT0.BIN
HEX file format (<I>INTEL(80/86) /<M>MOTOROLA(S1) /<T>TEKTRONICS) :I

Wait...
INTEL 80/86 Hex to Binary converter
Convert complete
```

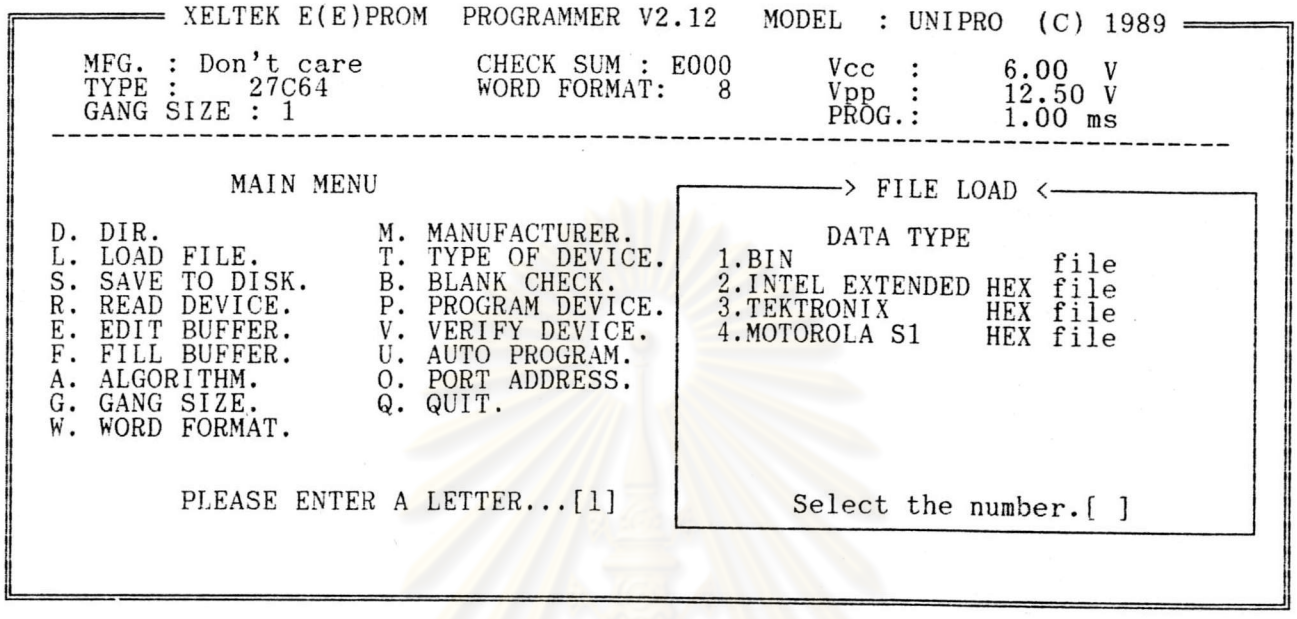
รูปที่ 7.31 แสดงการใช้งานโปรแกรม HEXOBJ

จากนั้นก็ถึงขั้นตอนการใช้โปรแกรม ROM212 เพื่อโปรแกรมอีพรอมซึ่งมีขั้นตอนดังนี้

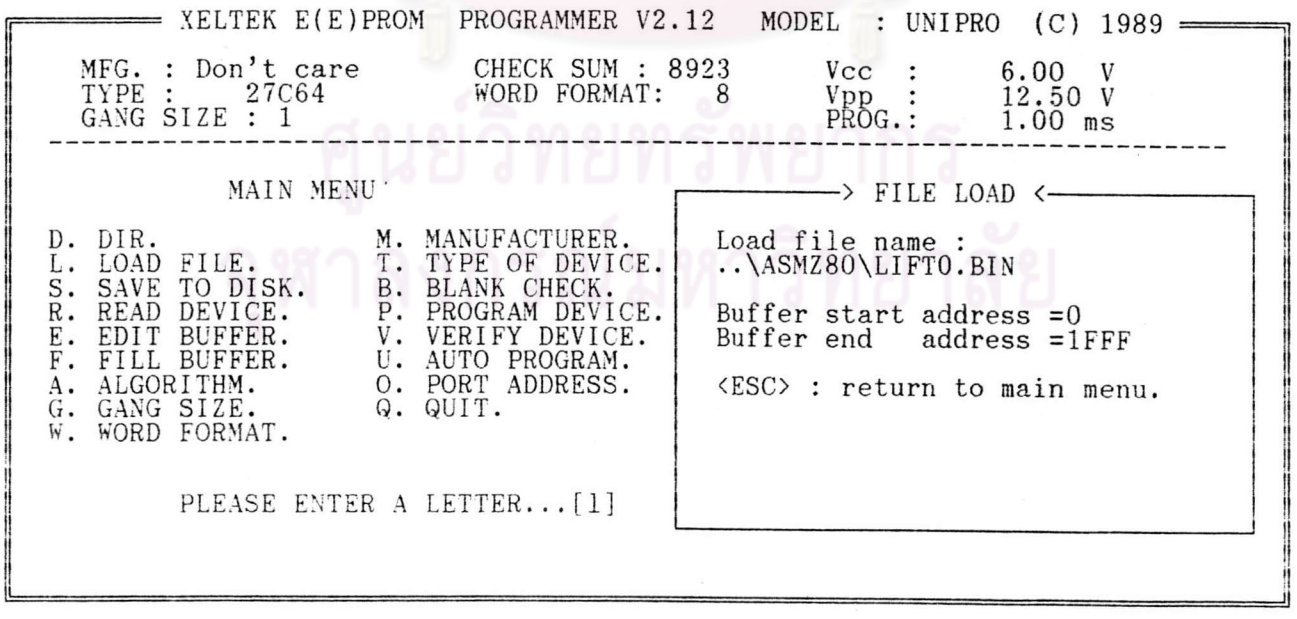
1. เข้าสู่ไดเรกตอรี UNIPRO และเปิดไฟเข้าเครื่องโปรแกรม UNIPRO
2. เรียกโปรแกรม ROM212 แล้วเลือกชนิดของอีพรอมที่จะโปรแกรม
3. กด L เพื่อโหลดไฟล์ลงสู่บัฟเฟอร์
4. จอภาพปรากฏดังรูปที่ 7.32 ให้กดหมายเลข 1 เลือกโหลดไฟล์แบบไบนารี
5. จอภาพปรากฏดังรูปที่ 7.33 ใส่ชื่อไฟล์ LIFT0.BIN
6. เสียบอีพรอมลงซอกเก็ตบนเครื่อง UNIPRO และทำการ Blank Check
7. กด P เพื่อสั่งโปรแกรมอีพรอม จอภาพจะปรากฏดังรูปที่ 7.34
8. หลังจากโปรแกรมเรียบร้อยแล้ว ให้เอาอีพรอมออกจากซอกเก็ตและปิดเครื่องโปรแกรม UNIPRO ก่อนออกจากโปรแกรม ROM212

ดังนั้นเราจะได้อีพรอมที่มีโปรแกรมควบคุมการทำงานของสิพต์จำลองไปใช้ในงานจริง โดยถอดบอร์ดอีมูเลชันออก เสียบบอร์ดซีพียู Z-80 ที่มีอีพรอมตัวที่โปรแกรมแล้วเข้ากับคอน

เนกเตอร์บนบอร์ดแม่ ประกอบวงจรตามที่ออกแบบไว้แล้วและเปิดไฟเข้าระบบพัฒนาไมโคร โปรเซสเซอร์ระดับบอร์ด ก็จะได้ระบบสิฟต์จำลองที่ทำงานได้ตามต้องการดังรูปที่ 7.35



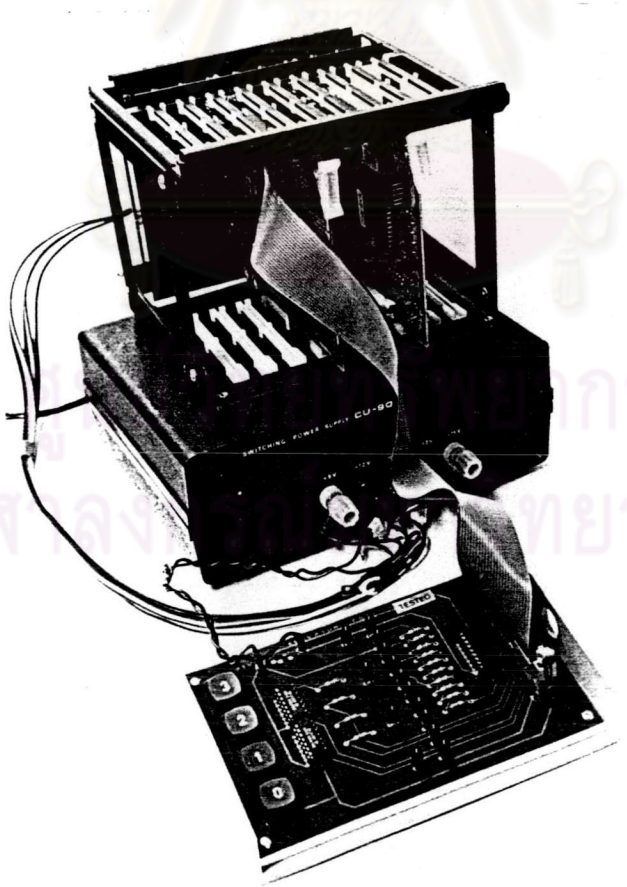
รูปที่ 7.32 แสดงจอภาพของโปรแกรม ROM212 เมื่อกด L



รูปที่ 7.33 แสดงจอภาพของโปรแกรม ROM212 ขณะโหลดไฟล์

```
XELTEK E(E)PROM  PROGRAMMER V2.12   MODEL  : UNIPRO  (C) 1989

MFG. : Don't care        CHECK SUM : 8923      Vcc  :     6.00  V
TYPE :    27C64          WORD FORMAT:   8      Vpp  :    12.50  V
GANG SIZE : 1                                  PROG.:     1.00  ms
-------------------------------------------------------------------

          MAIN MENU                          -----> ROM PROGRAM <-----

D. DIR.                 M. MANUFACTURER.    chip  start  :     0
L. LOAD FILE.           T. TYPE OF DEVICE.  chip  end    :  1fff
S. SAVE TO DISK.        B. BLANK CHECK.     buffer start :     0
R. READ DEVICE.         P. PROGRAM DEVICE.  buffer end   :  1fff
E. EDIT BUFFER.         V. VERIFY DEVICE.   Want to program(change):(y/c/ESC)
F. FILL BUFFER.         U. AUTO PROGRAM.
A. ALGORITHM.           O. PORT ADDRESS.
G. GANG SIZE.           Q. QUIT.
W. WORD FORMAT.

        PLEASE ENTER A LETTER...[p]
```

รูปที่ 7.34 แสดงจอภาพของโปรแกรม ROM212 เมื่อสั่งโปรแกรมอีพรอม



รูปที่ 7.35 แสดงระบบลิฟต์จำลองที่พัฒนาเรียบร้อยแล้ว