รายการอ้างอิง

ภาษาไทย

กองบรรณาธิการ ไมโครคอมพิวเตอร์. คน vs คอมพิวเตอร์. ไมโครคอมพิวเตอร์ ฉบับที่ 52 (กันยายน 2532): 221-229.

เกรียงศักดิ์ บุญเสริมสุรวงศ์. เมื่อคุณจะทำโครงงานทางด้านไมโคร. เซมิคอนดัคเตอร์อิเลคทรอนิคส์ ฉบับที่ 89 (พฤศจิกายน - ธันวาคม 2531): 212-218.

คู่มือไอซีไมโครโปรเซสเซอร์. กรุงเทพมหานคร: บริษัทซีเอ็ดยูเคชั่นจำกัด, 2529.

ทวี โชคพิพัฒน์. รายงาน Senior Project เรื่อง Z-80 Emulator. กรุงเทพมหานคร: ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2531.

บริษัทเมเชอร์โทรนิกซ์ จำกัด. ฟลุค 9010A คอมพิวเตอร์สำหรับซ่อมไมโครคอมพิวเตอร์และไมโครโปรเซสเซอร์. เซมิคอนดัคเตอร์อิเลคทรอนิคส์ ฉบับที่ 68 (ธันวาคม 2529-มกราคม 2530): 108-112.

บริษัท อีทีที จำกัด. อีที - บอร์ด 3.0. เซมิคอนดัคเตอร์อิเลคทรอนิคส์ ฉบับที่ 93 (กรกฎาคม 2532): 193 - 195.

พรชัย จิตต์พานิชย์. ปฏิบัติการเขียนโปรแกรมภาษาแอสแซมบลี้ภายใต้ ซีพีเอ็ม-80. กรุงเทพมหานคร: สำนักพิมพ์มหาวิทยาลัยรามคำแหง, 2531.

ยืน ภู่รวรรณ และวัฒนา เชียงกูล. ไมโครโปรเซสเซอร์ ไมโครคอมพิวเตอร์ (Z-80 Microprocessor). กรุงเทพมหานคร: บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2534.

วศิน เพิ่มทรัพย์. คู่มือ MS-DOS PC-DOS. กรุงเทพมหานคร: VS Enterprise, 2532.

วิริยะ กองรัตน์ และวิชัย ตันติจริยางกูร. ระบบช่วยพัฒนาไมโครโปรเซสเซอร์. ใน การประชุมวิชาการวิศวกรรมไฟฟ้า ครั้งที่ 10 (เล่ม 2). หน้า 2-439 - 2-447. กรุงเทพมหานคร: ภาควิชาวิศวกรรมไฟฟ้า คณะวิศกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2530.

เศรษฐา พันธุ์เพ็ง. รายงานการค้นคว้าประกอบวิชา 162499 เรื่อง Z-80 Emulator. กรุงเทพมหานคร: ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2532.

สินชัย กมลภิวงศ์, เดชา สุรัจกุลวัฒนา และสิทธิชัย กาญจนาธำรงค์กุล. การพัฒนาลอจิกอนาไลเซอร์ ขนาด 40 ช่องสัญญาณราคาถูก. ใน การประชุมทางวิชาการวิศวกรรมไฟฟ้า 9 สถาบัน ครั้งที่ 11 (เล่ม 1). หน้า 1-12-1 - 1-12-12. กรุงเทพมหานคร: ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมเทคโนโลยี สถาบันเทคโนโลยีราชมงคล, 2531.

เสกสัณห์ วัฒนะโชติ. ซอฟต์แวร์สำหรับอินเซอร์กิตอีมูเลเตอร์สำหรับไมโครโปรเซสเซอร์ Z-80 และ 8085. กรุงเทพมหานคร: ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2535.

ภาษาอังกฤษ

Advance Micro Devices. PALASM V2.23D Mkt [Computer program]. Santa Clara, CA: Advance Micro Devices, 1989.

Aseo, J. Advanced microprocessors demand high-level language Support. Digital Design (June,1985): 75-79.

Blakemore, D. Advance microprocessors require complex appropach. Computer Design (June 1,1987): 85-90.

Braun, J. and Chong, D. Portable development system blazes new trails. Electronics (June 2, 1982): 163-168.

Caine, Farber & Gordon, Inc. 80/DS experts-PL/M. Pasadena, CA: Caine, Farber & Gordon, Inc., 1986.

Clements, A. Microprocessor system design: 68000 hardware, software, and interfacing. Boston: PWS-KENT Publishing Co., 1987.

Coffron, J. W. Z80 applications. Berkeley: SYBEX Inc., 1983.

Datastrom Technologies, Inc. Procomm plus-version 1.1B [Computer program]. Datastrom Technologies, Inc., 1988.

Electro Systems Associates Pvt Ltd. ESA ICE-1 user's guide Bangalore India:Electro Systems Associates Pvt Ltd.

Falk, H. Development systems evolve toward integrated, host-indenpendent solutions. Computer Design (April 1, 1988): 44-50.

_____. Emulator set sights on ever-faster processor chips. Computer Design (May 1, 1988): 59-64.

Hall, D. V. Microprocessor and interfacing: Programming and hardware. New York: McGraw-Hill, Inc., 1988.

Harding, B. Logic analyzers and emulators evolve with the times. Computer Design (June 1, 1989):67-76.

Hordeski, M. F. Microprocessor in industry. P.E. Van Nostrand Reinhold, 1984.

IBM Corp. Personal computer XT hardware reference library Technical reference. Florida: IBM Corp., 1983.

Intel Corp. MCS-80 user's manual (with introduction to MCS-85). Santa Clara, CA:Intel Corp., 1977.

____. Microprocessor and peripheral handbook volume II Peripheral. Santa Clara, CA:Intel Corp., 1988.

Johnson, M. In-circuit emulation makes software development manageable. Computer Design (April 1,1986): 79-86.

Kline, B., Maerz, M. and Rosenfeld, P. The in-circuit approach to the development of microcomputer-based products. In W. C. Lin (ed.), Microprocessor: Fundamentals & applications. pp. 183-188. IEEE Press, 1976.

Leibson, S. H. In-circuit emulator for uCs. EDN (July 20, 1989): 64-78.

Lejine, B. In-circuit emulator I. In V. Tseng (ed.),Microprocessor development and development systems. New York: McGraw-Hill, Inc., 1982.

Leonard, M. In-circuit emulators crowd eagerly onto the PC platform. Electronic Products (February 1, 1987):19-22.

Microtec Research Inc. ASMZ80.CMP: Test program for Z80 relocatable macro assembler (Paragon ASMZ80 assembler - version 4.3D) [Machine-readable data file]. Microtec Research Inc., 1985.

____. CMP85.LST: Test program for 8080/8085 relocatable macro assembler (Paragon ASMZ80 assembler -version 5.3d) [Machine-readable data file]. Microtec Research Inc., 1987.

Microtek International Inc. MICE user's guide. Hsinchu, Taiwan: Microtek International Inc., 1984.

____. MICE-II user's guide - for 8-bit series microprocessors. Hsinchu, Taiwan: Microtek International Inc., 1984.

Mihalik, M. In-circuit emulator II. In V. Tseng (ed.),Microprocessor development and development systems. New York: McGraw-Hill, Inc., 1982.

Monolitic Memories.Palasm 2.

Multitech Industrial Corp. Micro-professor MPF-I user's and experiment manual. Taipei, Taiwan: Multitech Industrial Corp. , 1981.

Osborne, A. An introduction to microcomputers (vol. 2). Berkley, CA: Osborne & Assosiates, Inc., 1978.

Peatman, J. B. Microcomputer-based design. New York: McGraw-Hill, Inc., 1977.

Rafiquzzaman, M. Microprocessors and Microcomputer Development Systems. New York: Harper & Row, Publishers, Inc., 1984.

Ramesh B. S., Jamadagni H. S. and Marco Olgiati. Microprocessor laboratory primer. Bangalore, India: CEDT, 1987.

Riff, J. Z80 & CP/M 2.2 emulator v3.10 [Computer program]. McLean, VA: Computerwise Consulting services, 1986.

Robidoux, G. and Dmitroca, R. Logic analyzer. Radio electronics (June 1991): 31-87.

Short, K. L. Microprocessor and programmed logic. 2nd ed. New York: Prentice-Hall, Inc., 1987.

Slater, M. Microprocessor-based design A comprehensive guide to hardware design. New York: Prentice-Hall, Inc., 1989.

Small, C. H. ROM emulation reach far-flung fields. EDN (March 1, 1990): 57-62.

Wright, M. uP simulators let you debug software on an IBM PC. EDN (December 11, 1986): 196-204.

ZAX Corp. ICD-278 for Z80 user's manual. Irvine, CA: ZAX Corp.,1985.

ภาคผนวก

ภาคผนวก ก.

คู่มือการใช้งาน

ความนำ

อินเซอร์กิตอีมูเลเตอร์เป็นเครื่องมือที่สำคัญในการพัฒนาระบบที่ใช้ไมโครโปรเซสเซอร์ ในงานวิจัยนี้ใช้สำหรับพัฒนาไมโครโปรเซสเซอร์ Z-80 และ 8085 การใช้งานต่อกับคอมพิวเตอร์ผ่าน RS 232C โปรแกรมที่ใช้บนเครื่องคอมพิวเตอร์ คือ PROCOMM PLUS 1.1B

ข้อเด่นของอินเซอร์กิตอีมูเลเตอร์ในงานวิจัยนี้คือ

มีหน่วยความจำอีมูเลชันขนาด 64 กิโลไบต์

เลือกใช้และกำหนดรูปแบบของหน่วยความจำได้ด้วยคำสั่งช่วงละ 2 กิโลไบต์

มีการแสดงโปรแกรมในหน่วยความจำเป็นภาษาแอสเซมบลี

รับข้อมูลภาษาแอสเซมบลีแปลเป็นข้อมูลเก็บไว้ในหน่วยความจำได้

ควบคุมการรับหรือไม่รับสัญญาณควบคุมจากระบบเป้าหมายได้

มีการติดตามการทำงานของซีพียูในเวลาจริง และเก็บในบัฟเฟอร์ขนาด 2048 X 40 บิต โดยบันทึกสัญญาณภายนอกได้ 8 สัญญาณ

องค์ประกอบของระบบ

ในการใช้งานอินเซอร์กิตอีมูเลเตอร์ จำเป็นต้องมีอุปกรณ์ดังต่อไปนี้

1. เครื่องอินเซอร์กิตอีมูเลเตอร์สำหรับ Z80 หรือ 8085 ตามระบบเป้าหมาย
2. แหล่งจ่ายไฟตรง 5 โวลท์ 3 แอมป์ +/-12 โวลท์ 100 มิลลิแอมป์
3. IBM PC หรือเครื่องที่เข้ากันได้
4. โปรแกรม PROCOMM PLUS 1.1B
5. ระบบเป้าหมายที่จะพัฒนาใช้ไมโครโปรเซสเซอร์ Z80 หรือ 8085

การต่อสายของระบบ

1. ต่อสาย RS232C ระหว่าง IBM PC กับอินเซอร์กิตอีมูเลเตอร์ ใช้สายสำหรับต่อโมเดมกับ IBM PC โดยขั้วต่อของอินเซอร์กิตอีมูเลเตอร์ เหมือนกับโมเดมสายต่อชนิดนี้เป็นชนิดต่อตรงทุกสาย

2. สายต่อที่มีขั้วต่อ 40 ขา ระหว่างซ็อกเก็ตของซีพียูในระบบเป้าหมายระวังการเสียบต้องให้ขาหนึ่งอยู่ตรงตามที่กำหนด

3. ต่อสายแหล่งจ่ายไฟตรง 5 โวลท์ และ +/- 12 โวลท์ กับอินเซอร์กิตอีมูเลเตอร์

การเลือกใช้สัญญาณนาฬิกา

ที่หน้าปัดด้านหน้ามีดิปสวิทช์ 2 หลัก สำหรับเลือกว่าจะใช้สัญญาณนาฬิกาจากวงจรภายในหรือจากขาสัญญาณนาฬิกาจากระบบเป้าหมาย

การเลือกสัญญาณนาฬิกาจากภายใน สวิทช์ 1 ON สวิทช์ 2 OFF

การเลือกสัญญาณนาฬิกาจากภายนอก สวิทช์ 1 OFF สวิทช์ 2 ON

สำหรับ Z-80 เท่านั้น สามารถเลือก สวิทช์ 1 ON สวิทช์ 2 ON เพื่อเลือกใช้สัญญาณนาฬิกาจากภายในและให้ต่อไปใช้ในระบบเป้าหมายได้ ข้อควรระวังคือบนระบบเป้าหมายต้องไม่มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ด้วยมิฉะนั้นจะทำให้วงจรผลิตสัญญาณนาฬิกาในอินเซอร์กิตอีมูเลเตอร์เสียหายได้

การเปิดไฟเข้าเครื่อง

การเปิดไฟควรทำตามลำดับดังนี้

1. IBM PC
2. อินเซอร์กิตอีมูเลเตอร์
3. ระบบเป้าหมาย

การใช้โปรแกรม PROCOMM PLUS 1.1B

1. ในแผ่นโปรแกรม PROCOMM PLUS 1.1B มีแฟ้มข้อมูลที่จำเป็น 2 แฟ้ม คือ PCPLUS.EXE และ PCSETUP.EXE

2. เรียกใช้โปรแกรมโดยพิมพ์ PCPLUS กด Enter

3. บนจอภาพจะปรากฏรูปตัวอักษร PROCOMM+ ขนาดใหญ่ กดปุ่มใดๆ

4. ขั้นตอนนี้ทำเฉพาะเมื่อเริ่มใช้งานครั้งแรก

4.1 กด Alt-P จะได้ผลดังแสดงในรูป 8.1

4.2 กด 5 เพื่อเลือกอัตราการรับส่ง 9600 บิตต่อวินาที

4.3 รูปแบบการรับส่งที่ใช้กับอินเซอร์กิตอีมูเลเตอร์คือ 9600,N,8,1 แล้วดูข้อความในบรรทัดบนสุดของกรอบภาพให้ตรงตามนี้ ถ้าไม่ตรงกด Alt-N

4.4 กด Alt-S กรอบภาพจะหายไป

4.5 กด Alt-S อีกครั้งจะปรากฏกรอบภาพดังรูป 8.2

4.6 กดปุ่มลูกศรลงจนแถบสว่างเลื่อนมาที่ TERMINAL OPTIONS กด Enter 2 ครั้งจะปรากฏกรอบภาพตามรูป 8.3

4.7 กด C ต่อด้วย Space เพื่อให้ (XON / XOFF) ON กด Enter

4.8 กด Esc 2 ครั้ง จะกลับไปรูป 8.2

4.9 กดปุ่มลูกศรลงจนแถบสว่างเลื่อนไปที่ ASCII TRANSFER OPTIONS กด Enter จะปรากฏกรอบภาพตามรูป 8.4

4.10 กด D แล้วพิมพ์ 0 กด Enter

4.11 กด E แล้วพิมพ์ 0 กด Enter

4.12 กด F แล้วพิมพ์ 6 กด Enter

4.13 กด Esc จะกลับไปรูป 8.2

4.14 กดปุ่มลูกศรลง จนแถบสว่างเลื่อนไปที่ SAVE SETUP OPTIONS กด Enter

5. ใช้งานตามปกติตามที่แสดงในเรื่องคำสั่งการใช้งาน

6. เลิกการทำงาน กด Alt-X กด Enter

```
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:
```

รูป 8.1 การตั้งอัตรารับส่งข้อมูลสำหรับ PROCOMM PLUS

```
PROCOMM PLUS SETUP UTILITY                                  MAIN MENU

                        MODEM OPTIONS
                        TERMINAL OPTIONS
                        KERMIT OPTIONS
                        GENERAL OPTIONS
                        HOST MODE OPTIONS
                        ASCII TRANSFER OPTIONS
                        FILE/PATH OPTIONS
                        COLOR OPTIONS
                        PROTOCOL OPTIONS
                        SAVE SETUP OPTIONS

Alt-Z: Help |      Press   or   to select, <↵ to accept      | Esc: Exit
```

รูป 8.2 การเลือกรายการหลักของ PROCOMM PLUS

```
PROCOMM PLUS SETUP UTILITY                              TERMINAL OPTIONS

A- Terminal emulation ..................... ANSI

B- Duplex ................................. FULL

C- Software flow control (XON/XOFF) .. ON

D- Hardware flow control (RTS/CTS) ... OFF

E- Line wrap .............................. ON

F- Screen scroll .......................... ON

G- CR translation ......................... CR

H- BS translation ......................... DESTRUCTIVE

I- Break length (milliseconds) ........ 350

J- Enquiry (ENQ) .......................... OFF

Alt-Z: Help |     Press the letter of the option to change:     | Esc: Exit
```

รูป 8.3 การเลือกเทอร์มินอล

```
PROCOMM PLUS SETUP UTILITY                          ASCII TRANSFER OPTIONS

A- Echo locally ..................... NO     K- Strip 8th bit ..... NO

B- Expand blank lines ............ NO

C- Expand tabs ..................... YES

D- Character pacing (millisec).. 0

E- Line pacing (1/10 sec)..... 0

F- Pace character ............... 6

G- CR translation (upload) ..... NONE

H- LF translation (upload) ..... STRIP

I- CR translation (download) ... NONE

J- LF translation (download) ... NONE

Alt-Z: Help |     Press the letter of the option to change:     | Esc: Exit
```

รูป 8.4 การเลือกการรับส่งแฟ้มข้อมูลแบบแอสกี

รูปแบบคำสั่งใช้งาน

คำสั่งต่างๆมีรูปแบบดังนี้

คำสั่งเรียกดูรูปแบบคำสั่งทั้งหมด

Help	H

คำสั่งเกี่ยวกับการกำหนดการใช้หน่วยความจำ

Memory Map	MM [range [T|S][P|D][RW|RO]|[N]]

คำสั่งเกี่ยวกับหน่วยความจำ

Dump	D [address|range]
Enter	E address [list]
Fill	F range list
Compare	C range address
Move	M range address
Memory Test	MT range
Search	S range list

คำสั่งเกี่ยวกับอินพุตเอาต์พุต

Input	I address
Output	O address byte

คำสั่งเกี่ยวกับการจัดการข้อมูลเป็นภาษาแอสเซมบลี

Assemble	A [address]
Unassemble	U [address|range]

คำสั่งเกี่ยวกับการอ่านเขียนแฟ้มข้อมูลในรูปแบบ Intel hex

Load	L
Write	W range

คำสั่งควบคุมการรับสัญญาณควบคุมซีพียูจากระบบเป้าหมาย

Pin P

คำสั่งรีเซตซีพียู

reset X

คำสั่งเกี่ยวกับการแสดงและแก้ไขค่ารีจิสเตอร์

Register R [register]

คำสั่งเกี่ยวกับการทำงานในโปรแกรมของผู้ใช้ทีละคำสั่ง

Trace T [value|I|M]

คำสั่งเกี่ยวกับการกำหนดจุดหยุด

Breakpoint B [P[S address [value]|C]]|[H[S|C]]|[X[H|L|C]]

คำสั่งเกี่ยวกับการให้ซีพียูทำงานในเวลาจริงจนกว่าจะพบจุดหยุด

Go G [address]

คำสั่งเกี่ยวกับการติดตามการทำงานของซีพียูในเวลาจริง

Trace Realtime TR B|C|F

Trace History TH [range[range[F][R][W][I][O][A]]]

ในการใช้งานให้พิมพ์อักษรตัวย่อคำสั่งตามด้วยพารามิเตอร์ต่างๆโดยมีสัญลักษณ์ของพารามิเตอร์ดังนี้

[ ] สิ่งที่อยู่ในวงเล็บนี้เป็นตัวเลือกที่จะใช้หรือไม่ใช้ก็ได้

| สิ่งที่อยู่หน้าหรือหลังเครื่องหมายนี้ให้เลือกตัวใดตัวหนึ่ง

address ตัวเลขฐานสิบหกไม่เกิน 4 หลัก เป็นแอดเดรสของหน่วยความจำ

value ตัวเลขฐานสิบหกไม่เกิน 4 หลักใช้เป็นตัวนับจำนวนครั้งในการทำงาน 0 หมายถึง 65536 ครั้ง

range มี 2 ลักษณะคือ

address1 address2

โดย address2 ต้องมากกว่าหรือเท่ากับ address1

หรือ

address1 L value

จะได้ address2 = address1 + value - 1

โดย address2 ต้องมากกว่าหรือเท่ากับ address1

byte เลขฐานสิบหกไม่เกิน 2 หลัก

list ข้อมูลเป็นไบต์ได้ 1 ไบต์ หรือหลายไบต์ คั่นด้วย , หรือตัวว่าง และยังใช้เป็นตัวอักษรได้โดยอยู่ในเครื่องหมายคำพูด คู่ใดคู่หนึ่งคือ " " หรือ ' ' ถ้าเริ่มด้วยเครื่องหมายใดจะถือว่าหมดข้อความเมื่อพบเครื่องหมายนั้นโดยไม่สนใจอีกเครื่องหมายหนึ่ง เราจึงสามารถเขียนข้อความเหล่านี้ได้ " I'm a boy."

หรือ 'This "literal"is correct.'

สามารถใช้ข้อความปนกับข้อมูลเป็นไบต์ได้

register ชื่อรีจิสเตอร์ของแต่ละซีพียู ดูรายละเอียดในคำสั่ง R

พารามิเตอร์อื่นที่แสดงอักษรด้วยตัวพิมพ์ใหญ่ให้ใช้อักษรนั้นเป็นพารามิเตอร์

ตัวอักษรทั้งหมดสามารถใช้ได้ทั้งตัวพิมพ์เล็กและตัวพิมพ์ใหญ่ โดยมีความหมายเหมือนกันยกเว้นข้อความในเครื่องหมายคำพูด ใน list

คำสั่งต่างๆจะมีผลเมื่อกด Enter แล้วเท่านั้น

สามารถใช้ปุ่ม Back Space ลบตัวอักษรที่ป้อนไปแล้วได้

ในระหว่างพิมพ์คำสั่ง ถ้าต้องการยกเลิกกด Esc

ในการทำงานของบางคำสั่งจะแสดงผลจำนวนมากจนดูผลบนจอไม่ทัน สามารถใช้ปุ่ม Ctrl-Numlock หรือ Pause เพื่อหยุดจอชั่วคราวได้ และกดปุ่มใดๆเมื่อต้องการให้แสดงผลต่อ ถ้าต้องการหยุดการทำงานกด Esc

การใช้งานคำสั่ง

รายละเอียดของแต่ละคำสั่ง ได้แก่

คำสั่งเรียกดูรูปแบบตารางคำสั่งทั้งหมด

Help	H
>H	
Assemble	A [address]
Breakpoint	B [P[S address [value]|C]]|[H[S|C]]|[X[H|L|C]]
Compare	C range address
Dump	D [address|range]
Enter	E address [list]
Fill	F range list
Go	G [address]
Help	H
Input	I address
Load	L
Move	M range address
Memory Map	MM [range [T|S][P|D][RW|RO]|[N]]
Memory Test	MT range
Output	O address byte
Pin	P
Quit	Q
Register	R [registername]
Search	S range list
Trace	T [value|I|M]
Trace Realtime	TR B|C|F
Trace History	TH [range[range[F][R][W][I][O][A]]]
Unassemble	U [address|range]
Write	W range
reset	X

คำสั่งเกี่ยวกับการกำหนดการใช้หน่วยความจำ

Memory Map MM [range [T|S][P|D][RW|RO]|[N]]

คำสั่งนี้ ใช้แสดงและแก้ไขการกำหนดการใช้หน่วยความจำของอินเซอร์กิตอีมูเลเตอร์ ถ้าไม่ใส่พารามิเตอร์จะแสดงค่าที่กำหนดไว้เดิม ในช่วงแอดเดรสทั้งหมด ถ้ากำหนด range จะแสดงเฉพาะช่วงแอดเดรสที่กำหนดเป็นช่วงละ 2 กิโลไบต์ เช่น 0 ถึง 7FFH คือ 2 กิโลไบต์แรก ถ้ากำหนดเป็น 0 ถึง 8 ก็จะเป็นช่วง 2 กิโลไบต์แรกเช่นกัน

ถ้ากำหนดพารามิเตอร์อื่นต่อท้ายจะเป็นการกำหนดสถานะใหม่โดยตัวอักษรแต่ละตัวที่อยู่คู่กันใน [ ] จะใช้ร่วมกันไม่ได้ ความหมายของตัวอักษรมีดังนี้

T ใช้หน่วยความจำจากระบบเป้าหมาย

S ใช้หน่วยความจำจากภายในอินเซอร์กิตอีมูเลเตอร์

P กำหนดให้เป็นหน่วยความจำสำหรับเก็บทั้งโปรแกรมและข้อมูล

D กำหนดให้เป็นหน่วยความจำสำหรับเก็บเฉพาะข้อมูล

RW กำหนดให้เป็นหน่วยความจำชนิดอ่านเขียนได้

RO กำหนดให้เป็นหน่วยความจำชนิดอ่านอย่างเดียว

N กำหนดว่าช่วงแอดเดรสนั้นไม่มีหน่วยความจำ

หลังจากแก้ไขจะแสดงสถานะที่กำหนดใหม่ ในช่วงแอดเดรสนั้นทั้งหมด

ตัวอย่าง แสดงสถานะเริ่มต้นเมื่อเปิดไฟ

```
>MM
0000-FFFF 64k System Prog Read/Write
>
```

ตัวอย่าง แก้ไขการกำหนดการใช้หน่วยความจำของอินเซอร์กิตอีมูเลเตอร์ช่วง 8 กิโลไบต์แรกให้เป็นหน่วยความจำชนิดอ่านอย่างเดียว 2 กิโลไบต์ ต่อไปกำหนดให้เป็นหน่วยความจำสำหรับเก็บเฉพาะข้อมูล ที่เหลือกำหนดว่าไม่มีหน่วยความจำ

```
>MM 0 1fff ro
0000-1FFF  8k System Prog Read-Only
>MM 2000 27ff d
2000-27FF  2k System Data Read/Write
```

```
>MM 2800 ffff n

2800-FFFF 54k Nonexistent

>MM

0000-1FFF  8k System Prog Read-Only

2000-27FF  2k System Data Read/Write

2800-FFFF 54k Nonexistent

>
```

คำสั่งเกี่ยวกับหน่วยความจำ

Dump D [address|range]

คำสั่งนี้ใช้ดูค่าข้อมูลในหน่วยความจำของผู้ใช้โดยแสดงแอดเดรสเริ่มต้นของบล็อกต่อด้วยข้อมูลบรรทัดละ 16 ไบต์ ในรูปเลขฐาน 16 แล้วต่อด้วยข้อมูลในรูปตัวอักษรถ้าข้อมูลใดแทนด้วยตัวอักษรไม่ได้จะแสดงด้วยจุด

กำหนด address คือ แอดเดรสเริ่มต้น ที่จะดูค่าข้อมูล จะแสดง 128 ไบต์

ตัวอย่าง ต้องการดูค่าข้อมูลที่แอดเดรส 0

```
>d 0
0000 CD 20 00 A0 00 9A EE FE-1D F0 F4 02 26 2F 2F 03 . ..........&//.
0010 26 2F BC 02 26 2F 79 13-01 01 01 00 02 FF FF FF &/..&/y.........
0020 FF FF FF FF FF FF FF FF-FF FF FF FF E8 2E 4E 01 ..............N.
0030 60 32 14 00 18 00 D5 32-FF FF FF FF 00 00 00 00 `2.....2........
0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0050 CD 21 CB 00 00 00 00 00-00 00 00 00 00 20 20 20 .!...........
0060 20 20 20 20 20 20 20 20-00 00 00 00 00 20 20 20          .....
0070 20 20 20 20 20 20 20 20-00 00 00 00 00 00 00 00          ........
>
```

กำหนด range คือ ช่วงแอดเดรสที่จะดูค่าข้อมูล

ตัวอย่าง ต้องการดูค่าข้อมูลที่แอดเดรส 199h ถึงแอดเดรส 1d9h

```
>d 199 1d4
0190                                  00 00 00 00 00 00 53          ......S
01A0 01 00 00 00 00 00 00 00-00 00 00 00 00 00 00 54 ...............T
01B0 01 00 00 00 00 00 00 00-00 00 00 00 00 00 00 55 ...............U
01C0 01 00 00 00 00 00 00 00-00 00 00 00 00 00 00 56 ...............V
01D0 01 00 00 00 00                                  .....
>
```

ถ้าไม่กำหนดพารามิเตอร์จะเริ่มแอดเดรสแรกที่ค่าต่อจากที่แสดงไว้ครั้งก่อนจำนวน 128 ไบต์

ตัวอย่าง

```
>d
01D0                  00 00 00-00 00 00 00 00 00 00 57       ..........W
01E0 01 00 00 00 00 00 00 00-00 00 00 00 00 00 00 58 ...............X
01F0 01 00 00 00 00 00 00 00-00 00 00 00 00 00 00 59 ...............Y
0200 01 00 00 00 00 00 00 00-00 00 00 00 00 00 00 5A ...............Z
0210 01 00 00 00 00 00 00 00-00 00 00 00 00 00 00 5B ...............[
0220 01 00 00 00 00 00 00 00-00 00 00 00 00 00 00 5C ...............\
0230 01 00 00 00 00 00 00 00-00 00 00 00 00 00 00 5D ...............]
0240 01 00 00 00 00 00 00 00-00 00 00 00 00 00 00 5E ...............^
0250 01 00 00 00 00                                  .....
>
Enter           E address [list]
```

คำสั่งนี้ใช้แก้ไขค่าข้อมูลในหน่วยความจำ

ถ้ากำหนด list จะเป็นการใส่ข้อมูลใน list ไปที่หน่วยความจำเริ่มจากที่กำหนด adress

ตัวอย่าง

```
>e 1800 "This's a boy" 0 1 2 3 4 5
```

```
>d 1800 1 20
1800 54 68 69 73 27 73 20 61-20 62 6F 79 00 01 02 03 This's a boy....
1810 04 05 00 00 00 00 00 00-00 00 00 00 00 00 00 BB ...............
>
```

ถ้ากำหนดเฉพาะ adress จะใช้แก้ไขข้อมูลทีละไบต์ ถ้าไม่ต้องการแก้ไขให้กดแป้นอักขระว่างจะเป็นการแก้ไขข้อมูลแอดเดรสถัดไป ถ้าต้องการกลับแก้ไขแอดเดรสก่อนหน้ากดเครื่องหมายลบ (-) ถ้าต้องการเลิกการทำงานกด Enter ในขณะที่ไม่มีการแก้ไขข้อมูล

ตัวอย่าง

```
>e 1800
1800  54.
1801  68.-
1800  54.90
1801  68.91
1802  69.
>d 1800 1 10
1800 90 91 69 73 27 73 20 61-20 62 6F 79 00 01 02 03 ..is's a boy....
>
```

Fill F range list

คำสั่งนี้ใช้ป้อนค่าข้อมูลใน list ลงสู่หน่วยความจำ ถ้า range ที่กำหนดยาวกว่าข้อมูลใน list ข้อมูลจะถูกใส่ซ้ำๆกันจนกว่าจะเต็ม ถ้า range สั้นกว่าข้อมูลที่เกินจะถูกตัดทิ้ง

ตัวอย่าง

```
>f 1800 1 20 0
>d 1800 1 20
1800 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ...............
1810 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ...............
>f 1800 1 20 "ROM/RAM "
>d 1800 1 20
1800 52 4F 4D 2F 52 41 4D 20-52 4F 4D 2F 52 41 4D 20 ROM/RAM ROM/RAM
1810 52 4F 4D 2F 52 41 4D 20-52 4F 4D 2F 52 41 4D 20 ROM/RAM ROM/RAM
```

>

Compare C range address

เปรียบเทียบข้อมูลในช่วงที่ระบุด้วยrange กับ ข้อมูลในช่วงที่เริ่มต้นด้วย address ถ้าข้อมูลต่างกันจะแสดงผลที่แตกต่างให้เห็นถ้าเหมือนกันทั้งหมดจะไม่แสดงผลใดๆ

```
>c180018 1804
1801  4F  41  1805
1803  2F  20  1807
1805  41  4F  1809
1807  20  2F  180B
>
```

Move M range address

ย้ายข้อมูลจากหน่วยความจำช่วงหนึ่งไปยังอีกช่วงหนึ่ง

```
>m1800 1805 1900
>d1900110
1900 52 4F 4D 2F 52 41 00 00-00 00 00 00 00 00 00 CA ROM/RA.....
>
```

Memory Test MT range

ทดสอบอ่านเขียนหน่วยความจำในช่วงที่กำหนด เพื่อตรวจสอบว่าเป็นหน่วยความจำแรมที่ถูกต้องหรือไม่ถ้าไม่ถูกต้องจะแสดงแอดเดรสเริ่มต้นที่อ่านได้ไม่ถูกต้องจากที่เขียนไว้

```
>mt 0 17ff
RAM error at address 0000
>mt 1800 1fff
RAM OK
>
```

Search S range list

หาค่าที่กำหนดในlist ในช่วงที่กำหนดโดย range

```
>s1800 18ff "RAM"
1804
180C
1814
181C
>
```

คำสั่งเกี่ยวกับอินพุตเอาต์พุต

Input I portaddress

คำสั่งให้อ่านค่าจากพอร์ตแอดเดรสที่กำหนด

แอดเดรสของพอร์ตสำหรับ Z-80 จะมีได้ตั้งแต่ 0 ถึง FFFF

สำหรับ 8085 มีค่าได้ไม่เกิน FF

ตัวอย่าง

```
>i90
FF
>
```

Output O portaddress byte

คำสั่งให้ส่งค่าข้อมูลที่กำหนดออกไปที่พอร์ตแอดเดรสที่กำหนด

ตัวอย่าง

```
>o 90 67
>
```

คำสั่งเกี่ยวกับการจัดการข้อมูลเป็นภาษาแอสเซมบลี

Assemble A [address]

คำสั่งนี้ทำให้ผู้ใช้สามารถป้อนโปรแกรมภาษาแอสเซมบลีลงสู่หน่วยความจำ ทีละบรรทัดได้และจะทำการแปลเป็นภาษาเครื่องแสดงไว้ท้ายบรรทัดด้วย

ตัวอย่าง การป้อนโปรแกรมของ Z-80 โดยเริ่มจากแอดเดรส 1800H

```
0000  INC  BC                                   03
0001  INC  DE                                   13
0002  LD   A,90                                 3E90
0004  LD   I,A                                  ED47
0006  LD   (IX+00),00                          DD360000
000A  PUSH AF                                   F5
000B  CALL P,F2F3                               F4F3F2
000E  POP  AF                                   F1
000F  RET  P                                    F0
0010  RST  28                                   EF
0011  XOR  ED                                   EEED
```

Unassemble U [address|range]

คำสั่งให้แสดงข้อมูลในหน่วยความจำออกมาเป็นโปรแกรมภาษาแอสเซมบลี บนหน้าจอ หากไม่กำหนดแอดเดรสท้าย คำสั่งจะแสดงทีละ 11 บรรทัด ถ้าไม่กำหนดพารามิเตอร์จะเริ่มแอดเดรสแรกที่ค่าต่อจากที่แสดงไว้ครั้งก่อน

ตัวอย่าง

```
>u
0013 ECEBEA   CALL PE,EAEB
0016 E9       JP   (HL)
0017 E8       RET  PE
0018 E7       RST  20
0019 E6E5     AND  E5
001B E4E3E2   CALL PO,E2E3
001E E1       POP  HL
001F E0       RET  PO
0020 DF       RST  18
0021 DEDD     SBC  A,DD
0023 DCDBDA   CALL C,DADB
```

>

คำสั่งเกี่ยวกับการอ่านเขียนแฟ้มข้อมูลในรูปแบบ Intel hex

Load L

คำสั่งรับแฟ้มข้อมูลในรูปแบบ Intel hex จากเครื่องไมโครคอมพิวเตอร์ ไปยังหน่วยความจำของระบบที่พัฒนา ด้วยคำสั่ง L

หลังจากนั้นใช้ PROCOMM PLUS ในการส่งข้อมูล ใช้คำสั่ง Upload ของโปรแกรม PROCOMM PLUS (กดปุ่ม Page Up) จะได้ผลดังแสดงในรูป 8.5 กด 4 แล้ว Enter เพื่อส่งแฟ้มรูปแบบ Intel hex ซึ่งเป็นแบบ ASCII จะได้รูป 8.6 ใส่ชื่อแฟ้มข้อมูล แล้ว Enter

ตัวอย่าง

```
>L
Use load file to COM utility of your terminal.
< Page Up >
Load ready
>
```

ในระหว่างที่ PROCOMM PLUS กำลังส่งข้อมูล ถ้าเกิดข้อผิดพลาดให้กดปุ่ม Esc เท่านั้น ห้ามกดปุ่มอื่นอาจทำให้ PROCOMM PLUS หยุดทำงานได้

Write W range

คำสั่งส่งข้อมูลจากหน่วยความจำในช่วงแอดเดรสที่กำหนดลงสู่ดิสค์ของเครื่องไมโครคอมพิวเตอร์ เป็นแฟ้มข้อมูลในรูปแบบ Intel hex

ตัวอย่าง

```
>w 1800 18ff
Use write com to file utility of your terminal.
strike a key when ready..
```

ช่วงนี้ใช้คำสั่ง Download (กดปุ่ม Page Down) ของโปรแกรม PROCOMM PLUS ในรูปแบบของ ASCII (Intel Hex) แล้วตั้งชื่อแฟ้มข้อมูลที่จะเก็บ เสร็จแล้วกดปุ่มใดๆ

อินเซอร์กิตอิมูเลเตอร์จะส่งข้อมูลในรูปแบบ Intel hex บนจอภาพจนจบจะได้

```
Write ready
>
```

```
PROCOMM PLUS Ready!

 Upload Protocols
 1) XMODEM        5) TELINK         9) WXMODEM       13) YMODEM-G BATCH
 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

 Your Selection:      (or press ENTER for XMODEM)
```

รูป 8.5 การส่งข้อมูลจาก PROCOMM PLUS

```
PROCOMM PLUS Ready!

 ASCII UPLOAD
 Please enter filename:
```

รูป 8.6 การใส่ชื่อแฟ้มข้อมูล

จากนั้นกดปุ่ม Esc

คำสั่งควบคุมการรับสัญญาณควบคุมซีพียูจากระบบเป้าหมาย

Pin P

สัญญาณควบคุมซีพียูจากระบบเป้าหมายแบ่งเป็น 4 กลุ่ม คือ

RESET (Z-80,8085)

BUSRQ (Z-80) หรือ HOLD (8085)

NMI (Z-80) หรือ TRAP (8085)

INT (Z-80) หรือ INTR, RST5.5, RST6.5, RST7.5

การใช้งานจะแสดงการกำหนดเดิมและให้แก้ไขด้วยตัวอักษร D (Disable) หรือ E (Enable)

ตัวอย่าง แสดงสถานะเริ่มต้นเมื่อเปิดไฟ (8085)

```
>P
        RESET   HOLD    TRAP    INT     (Enable/Disable)
        D       D       D       D
        E               E

>P
        RESET   HOLD    TRAP    INT     (Enable/Disable)
        E       D       E       D

>
```

คำสั่งรีเซตซีพียู

reset X

คำสั่งรีเซตให้ซีพียูกลับไปทำงานที่แอดเดรส 0000H

```
>X

>
```

คำสั่งเกี่ยวกับการแสดงและแก้ไขค่ารีจิสเตอร์

Register R [register]

ชื่อของรีจีสเตอร์ สำหรับ Z-80 ได้แก่

PC ,AF ,BC ,DE ,HL ,SP ,IX ,

IY ,AF',BC',DE',HL',A ,F ,B ,C ,D ,E ,H ,L ,I

IFF = Interrupt enable Flip Flop (IFF2)

SF = Sign Flag

ZF = Zero Flag

HF = Haft-carry Flag

HC = Haft-Carry Flag

PF = Parity/overflow Flag

NF = subtract flag

CF = Carry Flag

สำหรับ 8085 ได้แก่

PC ,AF ,BC ,DE ,HL ,SP ,A ,F ,B ,C ,D ,E ,H ,L

IM = Interrupt Mask

IE = Interrupt Enable

SF = Sign Flag

ZF = Zero Flag

AC = Auxiliary Carry

PF = Parity Flag

CF = Carry Flag

หากคำสั่งนี้ไม่มีชื่อรีจีสเตอร์ต่อท้าย จะเป็นคำสั่งขอดูค่าในรีจีสเตอร์ทั้งหมดแต่ถ้า มีชื่อรีจีสเตอร์ต่อท้ายคำสั่งจะเป็นการแก้ไขค่าในรีจีสเตอร์ตัวนั้นโดยจะแสดงค่าเดิมให้ดูด้วย

ตัวอย่าง ต้องการดูค่าในรีจีสเตอร์ทั้งหมดของซีพียู 8085

```
>R
  PC   A  F(SZAPC) B  C  D  E  H  L IM  SP
 0000 00 00 00000 00 00 00 00 00 00 07 0000
>
```

ตัวอย่าง ต้องการตั้งค่ารีจีสเตอร์ PC ให้เป็น 1800H

```
>R PC
PC  0000
1800
>
```

คำสั่งเกี่ยวกับการทำงานในโปรแกรมของผู้ใช้ทีละคำสั่ง

Trace T [value|I|M]

ทำงานในโปรแกรมของผู้ใช้ทีละคำสั่งถ้ากำหนดพารามิเตอร์เป็น I จะแสดงคำสั่งที่กำลังจะทำงานในรูปแอดเดรส สถานะและข้อมูลออพโค้ด รอให้ผู้ใช้กดปุ่มใดๆยกเว้น Enter และ Esc จะทำงานในคำสั่งนั้นแล้วแสดงคำสั่งนั้นเป็นภาษาแอสเซมบลี แล้วแสดงคำสั่งต่อไปเพื่อรอการทำงานถ้าต้องการเลิกทำงานให้กด Enter หรือ Esc

ทำงานในโปรแกรมของผู้ใช้ทีละแมชชีนไซเคิลถ้ากำหนดพารามิเตอร์เป็น M จะแสดงคำสั่งที่จะทำงานในรูปแอดเดรส สถานะและข้อมูล รอให้ผู้ใช้กดปุ่มใดๆก็จะทำงานในแมชชีนไซเคิลนั้นแล้วแสดงแมชชีนไซเคิลต่อไป สำหรับแมชชีนไซเคิลสุดท้ายของคำสั่งจะแสดงคำสั่งเป็นภาษาแอสเซมบลีด้วย การเลิกทำงาน เหมือนการใช้พารามิเตอร์ I

อักษรย่อที่ใช้แทนสถานะของซีพียู มีดังนี้

F = opcode Fetch

R = memory Read

W = memory Write

I = Input

O = Output

A = interrupt Acknowledge

H = Halt

ถ้ากำหนดพารามิเตอร์เป็น value จะทำงานทีละคำสั่งแล้วแสดงค่ารีจีสเตอร์ที่ได้จากคำสั่งนั้นแล้วทำงานคำสั่งต่อไปจนครบจำนวนคำสั่งที่กำหนดไว้ ถ้าไม่กำหนดพารามิเตอร์จะทำงานแบบนี้เพียงคำสั่งเดียว หลังจากเลิกทำงานจะแสดงคำสั่งต่อไปในรูปภาษาแอสเซมบลีอีกหนึ่งคำสั่ง

ตัวอย่าง ทำงานในโปรแกรมของผู้ใช้ทีละคำสั่ง

```
>TI
0100  F  7E  LD   A,(HL)
0101  F  FD  LD   A,(IY-10)
0104  F  DD  LD   (IX-70),45
0108  F  DD
>
```

ตัวอย่าง ทำงานในโปรแกรมของผู้ใช้ทีละแมชชีนไซเคิล

```
>TM
0000  F  DD
0001  F  21
0002  R  00
0003  R  10  LD   IX,1000
0004  F  DD
0005  F  7E
0006  R  00
1000  R  00  LD   A,(IX+00)
0007  F  DD
0008  F  46
0009  R  10
1010  R  10  LD   B,(IX+10)
000A  F  DD
000B  F  4E
000C  R  7F
107F  R  C1  LD   C,(IX+7F)
000D  F  DD
>
```

ตัวอย่าง ทำงานทีละคำสั่งแล้วแสดงค่ารีจีสเตอร์

```
>T7
 PC                          A  F(SZAPC) B  C  D  E  H  L IM  SP
0000 00       NOP           00 00 00000 00 00 00 00 00 00 07 0000
0001 00       NOP           00 00 00000 00 00 00 00 00 00 07 0000
0002 00       NOP           00 00 00000 00 00 00 00 00 00 07 0000
0003 00       NOP           00 00 00000 00 00 00 00 00 00 07 0000
0004 3D       DCR  A        FF 00 00000 00 00 00 00 00 0F 07 0000
0005 89       ADC  C        FF 00 00000 00 00 00 00 00 A0 07 0000
```

```
0006 FF       RST  7          FF 00 00000 00 00 00 00 00 00 07 FFFE
0038 00       NOP
>T
 PC                            A  F(SZAPC) B  C  D  E  H  L IM  SP
0038 00       NOP             FF 00 00000 00 00 00 00 00 00 07 FFFE
0039 00       NOP
>
```

คำสั่งเกี่ยวกับการกำหนดจุดหยุด

```
Breakpoint     B [P[S address [value]|C]]|[H[S|C]]|[X[H|L|C]]
```

การกำหนดจุดหยุดในคำสั่งนี้มี 3 แบบ คือ

การกำหนดจุดหยุดเป็นแอดเดรสที่ซีพียูอ่านคำสั่ง 1 แอดเดรส

กำหนดให้หยุดเมื่อซีพียู HALT

กำหนดให้หยุดเมื่อสัญญาณภายนอกเป็นสถานะตรงที่กำหนด

ตัวอย่าง ดูว่ามีการกำหนดจุดหยุดเป็นแอดเดรสที่ซีพียูอ่านคำสั่งไว้ที่ใด

```
>BP
Break Point Address = 1800
>
```

หากไม่มีจะแสดงคำว่า "Break Point Clear"

ตัวอย่าง กำหนดจุดหยุดปฏิบัติการที่แอดเดรส 1809H

```
>BPS 1809
>BP
Break Point Address 1809   Passcount 1
>
```

ตัวอย่าง มีการกำหนด value ใช้เป็นตัวนับจำนวนครั้งในการผ่านแอดเดรสจุดหยุดก่อนให้หยุดทำงานจริงๆ (ถ้าไม่กำหนด value คือ 1 ครั้ง 0 หมายถึง 65536 ครั้ง)

```
>BPS 1809 7
Break Point Address 1809   Passcount 7
>
```

ตัวอย่าง คำสั่งใช้ยกเลิกจุดหยุดปฏิบัติการที่ได้ตั้งไว้แล้ว

```
>BPC

>BP

Break Point Clear
```

ตัวอย่าง กำหนดให้หยุดเมื่อซีพียู HALT

```
>BHS

>
```

ตัวอย่าง ดูการกำหนดให้หยุดเมื่อซีพียู HALT

```
>BH

Break when Halt

>
```

ตัวอย่าง ยกเลิกการกำหนดให้หยุดเมื่อซีพียู HALT

```
>BHC

>BH

Break Halt Clear

>
```

ตัวอย่าง กำหนดให้หยุดเมื่อสัญญาณภายนอกเป็นสถานะ 1

```
>BXH

>
```

ตัวอย่าง ดูการกำหนดให้หยุดเมื่อมีสัญญาณภายนอก

```
>BX

Break eXternal High

>
```

ตัวอย่าง กำหนดให้หยุดเมื่อสัญญาณภายนอกเป็นสถานะ 0

```
>BXL

>BX

Break eXternal Low

>
```

ตัวอย่าง ยกเลิกการกำหนดให้หยุดเมื่อมีสัญญาณภายนอก

```
>BHC
>BH
Break eXternal Clear
>
```

คำสั่งเกี่ยวกับการให้ซีพียูทำงานในเวลาจริงจนกว่าจะพบจุดหยุด

```
Go              G [address]
```

คำสั่งให้ซีพียูทำงานตามเวลาจริงในโปรแกรม โดยเริ่มที่แอดเดรสที่ผู้ใช้กำหนดต่อท้ายคำสั่ง แต่ถ้าไม่มี จะเริ่มที่แอดเดรสซึ่งค่า PC ชี้อยู่ เมื่อหยุดจะแสดงสถานะการหยุด คำสั่งที่กำลังจะทำงานในรูปแอดเดรส สถานะและข้อมูลออพโค้ด

ตัวอย่าง สั่งทำงานตามโปรแกรมตั้งแต่แอดเดรส 1800H หยุดโดยผู้ใช้กด Esc

```
>G 1800
USER BREAK
0008  F  46
>
```

สถานะการหยุดมีดังนี้

USER BREAK

BREAK WHEN HALT

ACCESS NON-EXISTENT

FETCH OPCODE FROM DATA MEMORY

WRITE TO READ ONLY MEMORY

BREAK EXTERNAL

BREAK POINT

คำสั่งเกี่ยวกับการติดตามการทำงานของซีพียูในเวลาจริง

```
Trace Realtime TR B|C|F
```

คำสั่งเกี่ยวกับการติดตามการทำงานในเวลาจริง มี 3 คำสั่ง คือ

Trace Backward

เก็บข้อมูลตั้งแต่เริ่มทำงานจนถึงหยุดทำงาน ถ้าข้อมูลเต็มจะเก็บทับข้อมูลในช่วงแรก ข้อมูลที่ได้ คือการติดตามการทำงานในช่วง 2048 แมชชีนไซเคิลสุดท้ายก่อนจะหยุดตามจุดหยุดที่ผู้ใช้ตั้งไว้

ตัวอย่าง

```
>TRB
BREAK POINT at step 07FE
Trace stop at step 07FF
0008  F  46
>
```

Trace Center

เก็บข้อมูลตั้งแต่เริ่มทำงานจนพบจุดหยุดแล้วเก็บข้อมูลต่ออีก 1024 แมชชีนไซเคิล จึงหยุดการทำงาน ถ้าข้อมูลเต็มจะเก็บทับข้อมูลในช่วงแรก ข้อมูลที่ได้คือการติดตามการทำงานในช่วง 2048 แมชชีนไซเคิลสุดท้าย

ตัวอย่าง

```
>TRC
BREAK POINT at step 03FF
Trace stop at step 07FF
0008  F  46
>
```

Trace Forward

ทำงานตามปรกติโดยไม่มีการบันทึกจนพบจุดหยุดจึงเริ่มบันทึก แล้วทำงานต่อจนบัฟเฟอร์เต็มแล้วหยุดการทำงาน ข้อมูลที่ได้ คือ 2048 ไซเคิล หลังพบจุดหยุด

ตัวอย่าง

```
>TRF
BREAK POINT at step 0000
Trace stop at step 07FF
0008  F  46
>
```

```
Trace History  TH [range[range[F][R][W][I][O][A]]]
```

คำสั่งแสดงข้อมูลที่บันทึกในบัฟเฟอร์ เป็นการทำงานในโปรแกรมของผู้ใช้ทีละแมชชีนไซเคิล แสดงในรูปแอดเดรส สถานะและข้อมูล สำหรับแมชชีนไซเคิลสุดท้ายของคำสั่งจะแสดงคำสั่งเป็นภาษาแอสเซมบลีด้วย

ข้อมูลที่บันทึกในบัฟเฟอร์มีขนาด 2048 แมชชีนไซเคิล ไม่สะดวกในการหาจุดที่ต้องการ จึงมีการกำหนด range จะแสดงเฉพาะช่วงแอดเดรสของบัฟเฟอร์ที่กำหนดในช่วง 2 กิโลไบต์ เช่น 0 ถึง 7FF คือข้อมูลที่บันทึกในบัฟเฟอร์ทั้งหมด ถ้ากำหนดเป็น 0 ถึง 7 ก็จะเป็นช่วง 8 แมชชีนไซเคิลแรก

ตัวอย่าง

```
>TH 0 7
000 0000  F  DD
001 0001  F  21
002 0002  R  00
003 0003  R  10  LD   IX,1000
004 0004  F  DD
005 0005  F  7E
006 0006  R  00
007 1000  R  00  LD   A,(IX+00)
>
```

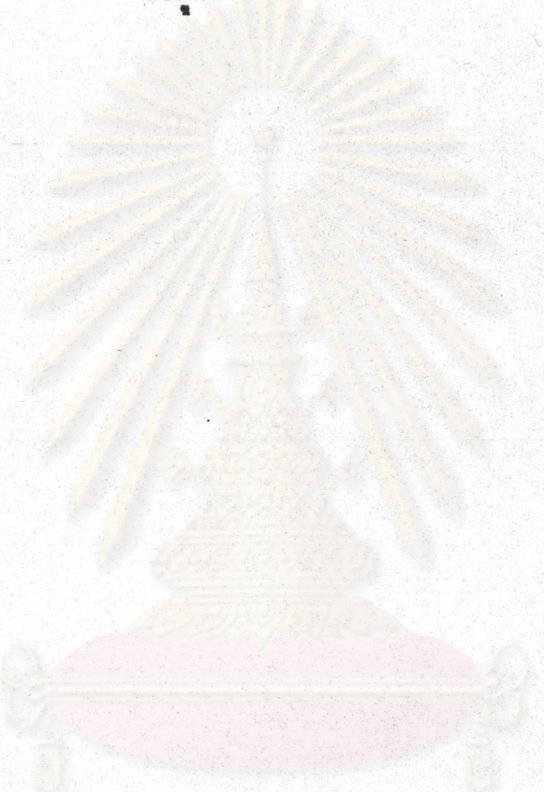
นอกจากนี้ยังมีการกำหนดแอดเดรสและสถานะของแมชชีนไซเคิลให้แสดงเฉพาะช่วงแอดเดรสและสถานะที่สนใจ ในช่วงแอดเดรสของบัฟเฟอร์ที่กำหนด

ตัวอย่าง ให้แสดง 8 แมชชีนไซเคิลแรก เฉพาะช่วงแอดเดรส 1000 ถึง 1100 และสถานะ R เท่านั้น

```
>TH 0 7 1000 1100 R
007 1000  R  00
>
```

ภาคผนวก ข.

ข้อมูลสำหรับการโปรแกรม PAL



```
TITLE    BREAK CONTROL
PATTERN  BRKCTRL.PDS
REVISION 1(RENAME)
AUTHOR   SEKSAN
COMPANY  EDL
DATE     16:44:55  2/27/1992

CHIP     BRKCTRL PAL16R4

MCSYNC  EMU     ACC_ER  WR_ER   FET_ER  BP      BO      BX      E_BP    GND
OE      E_BH    HALT    SRC0    SRC1    SRC2    BRK     E_BX    BXH     VCC

EQUATIONS

;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)
```

```
TITLE    CHIP SELECT
PATTERN  CS.PDS
REVISION 1(RENAME)
AUTHOR   SEKSAN
COMPANY  EDL
DATE     16:33:38  2/27/1992

CHIP     CS PAL10L8

C_A15   C_A14   C_A13   C_A12   C_A11   C_A10   C_A9    C_A8    C_IOM   GND
C_WR    CS_ROM  CS_8155 CS_BUS  CS_BRK  CS_COM  CS_MAP  CS_WRH  CS_WRL  VCC

EQUATIONS

/CS_ROM  = /C_A15
         * /C_IOM
;0000-7FFF  MEM

/CS_8155 =  C_A15*/C_A14*/C_A13*/C_A12*/C_A11
;8000-87FF

/CS_BUS  =  C_A15*/C_A14*/C_A13*/C_A12* C_A11*/C_A10
         *  C_IOM
;88  -8B    IO

/CS_BRK  =  C_A15*/C_A14*/C_A13*/C_A12* C_A11* C_A10
         *  C_IOM
;8C  -8F    IO

/CS_COM  =  C_A15*/C_A14*/C_A13* C_A12*/C_A11*/C_A10*/C_A9
         *  C_IOM
;90  -91    IO

/CS_MAP  =  C_A15*/C_A14*/C_A13*/C_A12*/C_A11* C_A10* C_A9*/C_A8
         *  C_IOM */C_WR
;86          IO WR

/CS_WRH  =  C_A15*/C_A14*/C_A13* C_A12*/C_A11* C_A10
         *  C_IOM
;94  -97    IO

/CS_WRL  =  C_A15*/C_A14*/C_A13* C_A12* C_A11*/C_A10
         *  C_IOM
;98  -9B    IO
```

```
TITLE    INTERRUPT CONTROL
PATTERN  INT85.PDS
REVISION 1
AUTHOR   SEKSAN
COMPANY  EDL
DATE     20:23:55  3/18/1992

CHIP     INT8085 PAL16L8

EMU     T_RSTI  E_RESET F_RESET T_TRAP  E_NMI   T_INTR  E_INT   T_RST75 GND
T_HLDA  INTR    TRAP    RST75   T_RST65 T_RST55 RST65   RST55   RSTI    VCC

EQUATIONS

/RSTI    = /T_RSTI* /E_RESET* EMU  ;ENABLE RESET FROM TARGET IN EMU
         + /F_RESET                ;FORCE RESET BY CONTROL CPU

/TRAP    = /E_NMI* EMU* /T_TRAP    ;ENABLE TRAP FROM TARGET 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

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

```
TITLE    INTERRUPT CONTROL
PATTERN  INTZ80.PDS
REVISION 1
AUTHOR   SEKSAN
COMPANY  EDL
DATE     11:25:33  11/13/1990

CHIP     INTZ80 PAL16L8

EMU     T_RESET E_RESET F_RESET T_NMI   E_NMI   T_INT   E_INT   M1      GND
T_BUSAK INT     NMI     T_M1    T_MREQ  T_IORQ  MREQ    IORQ    RESET   VCC

EQUATIONS

/RESET   = /T_RESET* /E_RESET* EMU ;ENABLE RESET FROM TARGET IN EMU
         + /F_RESET                ;FORCE RESET BY CONTROL CPU

 NMI     = /E_NMI* EMU* T_NMI      ;ENABLE NMI FROM TARGET WHEN 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

/T_MREQ  = /MREQ
 T_MREQ.TRST = T_BUSAK

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

```
TITLE    MEMORY READ/WRITE CONTROL
PATTERN  MEM85.PDS
REVISION 1
AUTHOR   SEKSAN
COMPANY  EDL
DATE     20:23:12  3/18/1992

CHIP     MEM8085 PAL16L8

RD       WR       EMU      ENABLE   IOM      A15      RSTO     SYSTEM   RW       GND
HLDA     E_HI     E_LO     T_HLDA   T_RD     T_WR     E_DATA   EM_RD    EM_WR    VCC

EQUATIONS

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

;READ AT TARGET ACTIVE WHEN
/T_RD      = /HLDA*                      ;BUSAK NOT ACTIVE AND
           ( /RD  * /IOM  * /EMU* ENABLE ;MEM_RD IN INTERR AND ENABLE(MEM)
           + /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(MEM)
           + /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(MEM)
           + /WR  * /EMU* /ENABLE        ;INTERR WRITE BUT NOT ENABLE(MEM)
           +  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(MEM)
           + /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(MEM)
           + /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
```

```
TITLE    MEMORY READ/WRITE CONTROL
PATTERN  MEMZ80.PDS
REVISION 4
AUTHOR   SEKSAN
COMPANY  EDL
DATE     21:43:22  11/10/1990

CHIP     MEMZ80 PAL16L8

RD       WR       EMU      ENABLE  MREQ     A15      RESET    SYSTEM  RW       GND
BUSAK    E_HI     E_LO     T_BUSAK T_RD     T_WR     E_DATA   EM_RD   EM_WR    VCC

EQUATIONS

;BUSAK AT TARGET ACTIVE WHEN
/T_BUSAK   = /BUSAK* EMU* ENABLE          ;BUSAK IN EMU AND ENABLE(BUSRQ)

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

```
TITLE    CHECK 8085 MACHINE CYCLE
PATTERN  SYNC85.PDS
REVISION 1
AUTHOR   SEKSAN
COMPANY  EDL
DATE     20:25:20  3/18/1992

CHIP     SYNC8085 PAL16L8

EXIST   RW      PROG    ALESYNC RD      ENWRBF  S0      S1      IOM     GND
INTA    MCSYNC  FETCH   WR      DIR     WRBF    FET_ER  WR_ER   ACC_ER  VCC

EQUATIONS

/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

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

 ACC_ER       = /EXIST* /IOM          ;ACCESS NON EXIST MEM

 WR_ER        = /RW   * /IOM* /S1*  S0 ;WRITE TO READ ONLY MEM

 FET_ER       = /PROG * /IOM*  S1*  S0 ;FETCH OPCODE FROM DATA MEM

/FETCH        =       /IOM*  S1*  S0
```

```
TITLE    CHECK OPCODE CB,ED,DD,FD
PATTERN  SYNCZ80.PDS
REVISION 3(RENAME)
AUTHOR   SEKSAN
COMPANY  EDL
DATE     16:39:19  2/27/1992

CHIP     SYNCZ80 PAL16L8

D4       HALT    D5       D6_3    D2       ENWRBF  D0_7    D1       M1       GND
IORQ     MCSYNC  RD       WR      DIR      WRBF    R_CBED  CBED     DDFD     VCC

;      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

EQUATIONS

/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

/DIR           = /RD              ;DATA BUS DIRECTION = IN
               + /M1 * /IORQ      ;WHEN READ OR INTAK

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

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

```
TITLE    TRACE
PATTERN  TRACE.PDS
REVISION 2(RENAME)
AUTHOR   SEKSAN
COMPANY  EDL
DATE     16:41:58  2/27/1992

CHIP     TRACE PAL16R4

WRCLK   CS_WRL  CS_WRH  C_AD4   C_AD5   C_AD6   C_AD7   C_WR    FUL     GND
OE      OVF     BFUL    ENBDLY  ENBFUL  ENWRBF  ENRDBF  WRHI    WRLO    VCC

EQUATIONS

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

/ENRDBF := /C_AD4
/ENWRBF := /C_AD5
/ENBFUL := /C_AD6
/ENBDLY := /C_AD7

;BFUL WORKS AS OPEN-COLLECTOR OUTPUT
BFUL.TRST = FUL * /ENBFUL
BFUL      = GND

;OVF WORKS AS RS-FF.
;FUL  HI SET FF.
;WRLO LO RESET FF.
;SET HAVE HIGHER PRIORITY
OVF     = FUL
        + OVF * WRLO
```

```
TITLE    WAIT CONTROL
PATTERN  WAIT85.PDS
REVISION 1
AUTHOR   SEKSAN
COMPANY  EDL
DATE     20:26:15  3/18/1992
CHIP     WAIT8085 PAL16L8
PIN 1  MCSYNC  ;HIGH NOT READ-WRITE CAN SET /WAIT, LOW CAN SET /BUSRQ
PIN 2  NEXT    ;CLOCK RISING EDGE DEACTIVATE /WAIT OR /BUSRQ
PIN 3  T_READY
PIN 4  F_WAIT  ;ACTIVE HIGH SET /WAIT WHEN NOT READ-WRITE
PIN 5  BRK     ;ACTIVE LOW FROM BRKCTRL
PIN 6  T_HOLD
PIN 7  ENABLE  ;ACTIVE HIGH ,ENABLE MEM IN /EMU, ENABLE BUSRQ IN EMU
PIN 8  EMU     ;ACTIVE HIGH I.E. EMULATION MODE
PIN 9  F_BUSRQ ;FORCE BUSRQ, ACTIVE LOW
PIN 10 GND
PIN 11 S0
PIN 12 READY
PIN 13 WAIT2   ;OUT WAIT1 AT NEXT RISING EDGE
PIN 14 WAIT1   ;LATCH F_WAIT WHEN NEXT LOW
PIN 15 S1
PIN 16 HALT
PIN 17 BUSRQ1  ;LATCH F_BUSRQ WHEN NEXT LOW
PIN 18 BUSRQ2  ;OUT BUSRQ1 AT NEXT RISING EDGE
PIN 19 HOLD
PIN 20 VCC
EQUATIONS
;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
/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
/HALT    = /S0* /S1
```

```
TITLE    WAIT CONTROL
PATTERN  WAITZ80.PDS
REVISION 1
AUTHOR   SEKSAN
COMPANY  EDL
DATE     23:01:56  11/19/1990
CHIP     WAITZ80 PAL16L8
PIN 1  MCSYNC  ;HIGH NOT READ-WRITE CAN SET /WAIT, LOW CAN SET /BUSRQ
PIN 2  NEXT    ;CLOCK RISING EDGE DEACTIVATE /WAIT OR /BUSRQ
PIN 3  T_WAIT
PIN 4  F_WAIT  ;ACTIVE HIGH SET /WAIT WHEN NOT READ-WRITE
PIN 5  BRK     ;ACTIVE LOW FROM BRKCTRL
PIN 6  T_BUSRQ
PIN 7  ENABLE  ;ACTIVE HIGH ,ENABLE MEM IN /EMU, ENABLE BUSRQ IN EMU
PIN 8  EMU     ;ACTIVE HIGH I.E. EMULATION MODE
PIN 9  F_BUSRQ ;FORCE BUSRQ, ACTIVE LOW
PIN 10 GND
PIN 11 NC
PIN 12 WAIT
PIN 13 WAIT2   ;OUT WAIT1 AT NEXT RISING EDGE
PIN 14 WAIT1   ;LATCH F_WAIT WHEN NEXT LOW
PIN 15 NC
PIN 16 NC
PIN 17 BUSRQ1  ;LATCH F_BUSRQ WHEN NEXT LOW
PIN 18 BUSRQ2  ;OUT BUSRQ1 AT NEXT RISING EDGE
PIN 19 BUSRQ
PIN 20 VCC
EQUATIONS
;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
```

ประวัติผู้เขียน

นายเสกสัณห์ วัฒนะโชติ เกิดวันที่ 8 ตุลาคม พ.ศ. 2508 ที่อำเภอสรรพยา จังหวัดชัยนาท สำเร็จการศึกษาปริญญาตรีวิศวกรรมศาสตรบัณฑิต สาขาอิเล็กทรอนิกส์ ภาควิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปีการศึกษา 2530 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต ที่จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ. 2531