ปีการศึกษา 2530

การพัฒนาเอดิเตอร์ภาษาไทย/ภาษาอังกฤษ

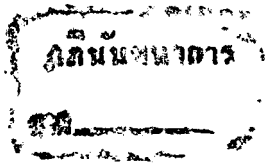
โดย

นาย เฉลิมชัย ลีชัยวัฒน์ 27-1042

นาย กิตติคุณ โพธิวนากุล 27-1023

อาจารย์ที่ปรึกษา

ดร. รัตติกร วรากูลศิริพันธุ์



ปริญญานิพนธ์ปีการศึกษา 2530

เรื่อง การพัฒนาเอดิเตอร์ภาษาไทย/ภาษาอังกฤษ

ผู้จัดทำ

1. เฉลิมชัย ลีชัยวัฒน์ 27-1042

2. กิตติคุณ โพธิวนากุล 27-1023

.....อาจารย์ที่ปรึกษา

(ดร. รัตติกร วรากูลศิริพันธุ์)

การพัฒนาเอดิเตอร์ภาษาไทย/ภาษาอังกฤษ

นาย เฉลิมชัย ลีชัยวัฒน์ 27-1042

นาย กิตติคุณ โพธิวนากุล 27-1023

อาจารย์ที่ปรึกษา

ดร. รัตติกร วรากูลศิริพันธุ์

ปีการศึกษา 2530

บทคัดย่อ

อิดิเตอร์ เป็นโปรแกรมที่ถูกสร้างมาเพื่อ เพื่อใช้ในการพิมพ์ข้อความ จดหมาย การเขียนโปรแกรม ใช้ในการเก็บข้อความ นอกจากนี้แล้วยังมีประโยชน์อื่นๆอีกมากมาย ลักษณะของอิดิเตอร์ในปริญญานิพนธ์นี้มีดังนี้

1 แสดงภาษาไทยและอังกฤษ 8 บรรทัด

2 สามารถที่จะเก็บข้อมูลลงในแผ่นดิสต์

3 สามารถที่จะเรียกดอสเชลได้

4 เป็นฟูลสกีนอิดิเตอร์หมายถึงสามารถที่จะเลื่อนเคอร์เซอร์ไปได้ทุกทิศทุกทาง

Introduction to THAI/ENGLISH EDITOR development

Chalermchai Leechaiwat 27-1042

Kittikun Potivanakul 27-1023

ADVISOR Dr. Rattikron Varakunsiripan

EDUCATION YEAR 1988

SHORT EPISODE

Editor is the programe that is used for typing letter creating document writing programe and it use for saveral ways.

the future of this editor is

1 It can display thai and english in 8 line per page.

2 It can save data to be a file in diskkete.

3 It can call OS cell so it can link with another software easily.

4 full screen editor is what you see what you get.

สารบัญ

บทที่ 1	บทนำ	1
บทที่ 2	ทฤษฎีและหลักการ	5
บทที่ 3	การคำนวนและการสร้าง	13
บทที่ 4	การทดลองและผลการทดลอง	28
บทที่ 5	สุปและวิจารณ์	35
ภาคผนวก		36

บทที่ 1

บทนำ

1.1 เอดิเตอร์คืออะไร

เอดิเตอร์เป็นอุปกรณ์ชนิดหนึ่ง ซึ่งใช้ช่วยในการจัดพิมพ์เอกสาร โดยช่วยให้สามารถแก้ไขเอกสารได้ง่าย พัฒนามาจากเครื่องพิมพ์ดีด ซึ่งมีปัญหาในการแก้ไขต้นฉบับ ซึ่งอาจเกิดจากความผิดพลาด หรือ ต้องการแก้ไขปรับปรุงสำนวนหรือข้อความบางตอน การใช้เอดิเตอร์จะทำให้การจัดการเอกสารเป็นไปได้อย่างสะดวกและมีประสิทธิภาพมากขึ้นโดยอาจมีชื่อเรียกต่าง ๆ กัน เช่น เวิร์ดโปรเซสเซอร์, โน๊ตเพด (note pad)

การใช้งานเอดิเตอร์ในปัจจุบันมีเครือข่ายกว้างกว่าเดิมมากอาทิเช่น ใช้ในการเขียนโปรแกรม (ซึ่งมีโปรแกรมแปรภาษาในปัจจุบันมักจะแถมมาด้วยเลย เช่น ในเทอร์โบปาลคาล) การใช้งานเอดิเตอร์ในการสั่งงานคอมพิวเตอร์เช่น ในเอ็มเอสดอสไลนคอมมานด์ ก็จะมีไลน์เอดิเตอร์ในการที่จะรับคำสั่งต่าง ๆ เข้าไป เช่น คำสั่งขอดูชื่อแฟ้มข้อมูล (dir) นอกจากนี้ในการป้อนภาษามนุษย์ในคอมพิวเตอร์เช่น เครื่องแปลภาษาที่ในการป้อนผ่านแป้นพิมพ์ ก็ย่อมจำเป็นต้องใช้เอดิเตอร์ทั้งสิ้น

1.2 ชนิดของเอดิเตอร์

อิดิเตอร์นั้นอาจแบ่งได้หลายชนิด อาทิเช่น

1.2.1. ไลน์เอดิเตอร์ (line editor) เป็นเอดิเตอร์ที่สามารถทำการแก้ไขได้เพียงทีละบรรทัดเท่านั้น ทำให้การแก้ไขข้อมูลทำได้ลำบาก เอดิเตอร์ชนิดนี้ง่ายต่อการสร้างที่สุด แต่ใช้งานได้ยากมาก ไม่สะดวก สำหรับเอดิเตอร์ชนิดนี้มักใช้ในการป้อนคำสั่งให้คอมพิวเตอร์ทีละคำสั่ง เช่น ในดอส ในแอปเปิลซอฟท์ เป็นต้น

1.2.2 ฟูลสกรีนเอดิเตอร์ (full screen editor) เป็นเอดิเตอร์ที่ใช้ในได้ง่ายกว่าแบบแรก เนื่องเราสามารถเห็นลักษณะของเอกสารที่เราสร้างได้ขณะแก้ไขโดยตรง เอดิเตอร์ชนิดนี้มีลักษณะคล้ายกับการเขียนลงบนกระดาษโดยตรง บางครั้งเราเรียกเอดิเตอร์แบบนี้ว่า เอดิเตอร์แบบสิ่งที่คุณเห็นเป็นสิ่งที่คุณได้(what you see what you get)

1.2.3 เอดิเตอร์ชนิดฉลาด (intelligent editor) เป็นชนิดย่อยๆของแบบที่สอง แต่มีความสามารถอื่น ๆ ประกอบด้วย อาทิเช่น มีความสามารถในการแก้คำผิดได้เอง โดยที่ต้องเอดิเตอร์จะมีการเก็บข้อมูลคำศัพท์ไว้ เมื่อมีศัพท์คำไหนที่ไม่ปรากฏในแฟ้มข้อมูลของตัวเองก็จะถามผู้ใช้ว่าผิดรึไม่ ถ้าไม่ผิดเป็นคำศัพท์ใหม่ก็จะบันทึกลงในแฟ้มของตัวเอง เอดิเตอร์แบบนี้มีน้อยและอยู่ในระหว่างวิจัยเท่านั้น

1.3 ประโยชน์ของเอดิเตอร์

โดยหลักแล้วก็คือ การรับข้อมูลเข้าคอมพิวเตอร์ แต่สามาถแยกให้เห็นเด่นชัดได้ดังนี้

1.3.1 ใช้ในการสั่งงานคอมพิวเตอร์

โดยมากใช้ไลน์เอดิเตอร์ เช่น ในดอส ในดีเบส

1.3.2 ใช้ในการเขียนโปรแกรม

โดยมาใช้ฟูลสกรีนเอดิเตอร์ ใช้ในการเขียนลำดับขั้นของคำสั่งในการสั่งงานให้คอมพิวเตอร์ มีปะโยชน์มากกว่าแบบแรก เพราะในทางปฏิบัติแล้วการสั่งงานที่มีความซับซ้อนให้คอมพิวเตอร์นิยมใช้วิธีนี้มากกว่า

1.3.3 การป้อนข้อมูลเข้าคอมพิวเตอร์

เช่นในระบบฐานข้อมูล เช่นชื่อที่อยู่ ซึ่งเราอาสัยอยู่ในเมืองไทยเราต้องใช้เอดิ

เตอร์ภาษาไทย

1.3.4 อิเลคโทรนิคส์เมล(electronic mail)

เป็นระบบการสื่อสารข้อมูลในรูปจดหมายจากผู้หนึ่งไปยังอีกผู้หนึ่ง โดยผ่านตัวกลางที่เป็นอิเลคโทรนิคส์ ซึ่งในที่นี้คือ คอมพิวเตอร์ การป้อนจดหมายเข้าไปในคอมพิวเตอร์นั้นจำเป็นต้องมีโปรแกรมส่วนเอดิเตอร์ทำหน้าที่รับข้อมูลก่อน

1.3.5 ระบบแปลภาษา

ระบบแปลภาษาจากภาษาไทยเป็นภาษาอื่นก็จำเป็นต้องใช้ เอดิเตอร์และต้องเป็นภาษาไทยด้วย ซึ่งระบบแปลภาษานี้มีแนวโน้วว่าจะมีการนำมาใช้จริงอย่างแน่นอน ในปัจจบันนี้ก็มีการวิจัยในเรื่องนี้อย่างหนัก

1.4 จุดประสงค์ของการพัฒนาเอดิเตอร์ภาษาไทย

1.4.1 ศึกษาระบบการรับและแสดงข้อมูลเป็นภาษาไทย

1.4.2 เพื่อเป็นพื้นฐานรองรับการพัฒนาระบบขึ้นในลำดับขั้นต่อไปคือ

1.4.2.1 อิเลคโทรนิคส์เมล โดยจะนำไปประกอบโครงข่ายคอมพิวเตอร์เพื่อการศึกษา (Education Computer Network ECN) ต่อไป เพื่อให้ระบบโครงข่ายดังกล่าวสมบูรณ์แบบยิ่งขึ้น

หมายเหตุ ระบบโครงข่ายดังกล่าวเป็นโครงการ ในหลักสูตรวิศวกรรมศาสตร์บัญฑิต โดยอยู่ในระหัสวิชา 14420 โครงการ 1 ของตัวนักศึกษาเอง โดยมี อาจารย์ รัตติกร เป็นอาจารย์ที่ปรึกษาเช่นเดียวกัน

1.4.2.2 ระบบแปลภาษา ซึ่งจะใช้ในการรับข้อมูลและแสดงผลข้อมูลส่วนที่เป็นภาษาไทย

สรุป

สำหรับในประเทศไทยนั้น ในปัจจุบันมีการตื่นตัวทางด้านเทคโนโลยีคอมพิวเตอร์มาก แต่การสั่งงานคอมพิวเตอร์ในปัจจุบันนั้น ยังเป็นภาษาอังกฤษอยู่ เอดิเตอร์ภาษาไทยถูกมองในรูปการจัดพิมพ์เอกสารในสำนักงานเท่านั้น ถ้ายังไม่มีการตื่นตัวในเรื่องนี้ การใช้งานคอมพิวเตอร์ในเมืองไทยจะถูกจำกัด เนื่องจากผู้ในต้องมีความรู้อย่างน้อยก็ต้องรู้จักภาษาอังกฤษ ยิ่งถ้ามีคอมพิวเตอร์รุ่นที่ 5 ซึ่งอาจใช้ภาษามนุษย์หรือคิดเองได้ด้วยแล้ว การสั่งงานด้วยภาษาไทยจะเป็นไปได้หรือ ? ถ้าแม้แต่การป้อนภาษาไทยยังทำได้ลำบาก โครงการนี้เป็นการศึกษาและพัฒนาการเรื่องดังกล่าว ก่อนที่จะก้าวไปสู่เอดิเตอร์แบบฉลาดหรืออิเลคโทรนิคส์เมลต่อไป

บทที่ 2

ทฤษฎีและหลักการ

2.1 การเขียนโปรแกรมโครงสร้าง

เนื่องจากเอดิเตอร์ที่พัฒนาขึ้นเป็นซอฟท์แวร์ ซึ่งในปัจจุบันนิยมการเขียนที่เป็นระบบที่เรียกว่า การเขียนโปรแกรมแบบโครงสร้าง (modular programing) ซึ่งทำให้การเขียนโปรแกรมเป็นไปอย่างรวดเร็วและมีประสิทธิภาพ

การเขียนโปรแกรมแบบโครงสร้างมีกฎทั่วไปดังนี้

2.1.1 แบ่งปัญหาออกเป็นขั้นตอนย่อย ๆ เราเรียกว่าโมดูล(module) โดยแต่ละโมดูลอิสระต่อกัน เว้นแต่การส่งค่าที่จำเป็น

2.1.2 จุดเข้าและจุดออกมีเพียงจุดเดียว (single entry single exit)

2.1.3 พยายามหลีกเลี่ยงความซับซ้อน ถ้าจุดไหนที่มีความยุ่งยากควรจะอธิบายในจุดนั้นให้ละเอียดว่า ทำอะไรบ้างและทำไมต้องทำอย่างนั้น

2.1.4 ซ่อนรายละเอียด จำกัดรายละเอียดข้อปลีกย่อยต่างๆที่อาจซับซ้อนไว้ในโมดูล อย่าค่าต่าง ๆ ในการคำนวนเกี่ยวเนื่องกับจนอาจทำให้เกิดความสลับซับซ้อนไปทั้งโปรแกรมได้

2.1.5 ควรแบ่งแยกชนิดตัวแปรที่ใช้เฉพาะในโมดูลออกจากตัวแปรหลักที่เกี่ยวเนื่องกันทั้งระบบ

2.1.6 คำนึงถึงความผิดพลาดที่อาจเกิดขึ้น และการแสดงระหัสความผิดพลาดกับไปยังส่วนที่เรียกมา

นอกจากนี้ หนังสือ ปีเตอร์ นอร์ตัน ภาษาแอสแซมบลี้ (Peter Norton'& Assembly

Language Book) ได้อธิบายหลักการเพิ่มจากที่กล่าวมาดังนี้

2.1.7 เก็บรักษาแล้คืนค่าในรีจีสเตอร์ต่าง ๆ ทุกตัว,ยกเว้นรีจีสเตอรตัวที่ใช้ในการผ่านค่าเท่านั้น

2.1.8 กำหนดหน้าที่ที่ใช้ในรีจีสเตอร์ที่เราใช้ในการผ่านค่าต่าง ๆเช่น

* ดีแอล (DL),ดีเอ็กซ์(DX) ใช้ในการผ่านค่าเป็นไบท์และเวิดส์

* ซีเอ็กซ์ (CX) ใช้ในการกำหนดจำนวนครั้งในการทำส่วนนั้น เป็นต้น

2.1.9 กำหนดและเขียนค่าต่าง ๆที่มีผลกระทบกับส่วนภายนอก

* ข้อมูลที่ต้องการป้อนเข้าไปในโมดูล

* ผลลัพธ์ที่จะส่งคืน

* การเรียกใช้โมมูลนั้น

* ตัวแปรที่ใช้ในโมดูล

2.2 โมดูลในเอดิเตอร์

โมดูลในเอดิเตอร์อาจแบ่งได้เป็นโมดูลใหญ่ใหญ่ ๆ ดังนี้

2.2.1 ส่วนจัดการเกี่ยวกับรับข้อมูลเป็นภาษาไทย

2.2.2 ส่วนจัดการเกี่ยวกับการแสดงผลเป็นภาษาไทย

2.2.3 ส่วนจัดการโครงสร้างข้อมูลในหน่วยความจำ

2.2.4 ส่วนควบคุมและจัดการ

2.3 ส่วนควบคุมและจัดการ

เป็นส่วนที่ทำหน้าที่รับข้อมูลทางแป้นพิมพ์และทำการแยกว่าเป็นข้อมูลหรือคำสั่งจากนั้นจะเป็นตัวส่งผ่านการควบคุมไปรายละเอียดปลีกย่อยของคำสั่งนั้น ๆ

2.4 การจัดการโครงสร้างของหน่วยความจำ

เมื่อเรารับข้อมูลมาแล้วเราก็จำเป็นต้องจัดการโครงสร้างของมันให้เป็นระเบียบ อาจเก็บได้หลายแบบดังนี้

2.4.1 การเก็บข้อมูลต่อเนื่องกันไปโดยขั้นระหว่างบรรทัดด้วยระหัสรีเทิรน์ (carriage return)และไลน์ฟีด(line feed) ซึ่งโครงแบบนี้เป็นโครงเดียวกับแฟ้มตัวอักษร (text file) ข้อได้เปรียบที่เห็นได้ชัดคือเก็นใส่จานแม่เหล็ก (diskette) ง่าย เพราะมีโครงสร้างแบบเดียวกัน ข้อเสียก็คือทำการแทรกข้อมูลหรือการลบข้อมูลทำได้ยาก หรือทำได้ช้า เนื่องจากจะต้องทำการเลื่อนข้อมูลถัดไปเรื่อย ๆเป็นจำนวนครั้งเท่ากับข้อมูลที่อยู่ทางขวาทั้งหมด การเข้าถึงข้อมูลยากเมื่อเป็นชนิดฟลูสกรีนเอดิเตอร์เนื่องโครงสร้างของข้อมูลที่จอภาพกับที่หน่วยความจำแตกต่างกัน ขนาดของแฟ้มข้อมูลที่สร้างด้วยโครงสร้างข้อมูลแบบนี้มักจะมีขนาดเล็กเนื่องจากความซ้ำเมื่อมีขนาดข้อมูลที่ใหญ่ จุดเด่นที่สำคัญของโครงสร้างข้อมูลแบบนี้คือประหยัดหน่วยความจำมาก

ตัวอย่างการเก็บของข้อมูลชนิดนี้

เราจะเป็นประโยค 2 ประโยคอยู่คนละบรรทัดว่า

เราเป็นนักศึกษา

และเราจะเป็นวิศวกร

การเก็บข้อมูลจะเป็น

เราเป็นนักศึกษา ,13,10, และเราจะเป็นวิศวกร

หมายเหตุ เลข 13 และเลข 10 เป็นระหัสแอสกี้ของรีเทิรน์และไลน์ฟีดตามลำดับ

2.4.2 โครงสร้างแบบอาเรย์ของบรรทัด โครงสร้างข้อมูลแบบนี้มีโครงสร้างที่

ง่ายต่อการเขียนโปรแกรมเพราะเป็นเสมือนตารางสี่เหลี่ยมเราสามารถที่จะเข้าถึงข้อมูลได้ง่ายโดยใช้สมการคณิตศาสตร์ง่าย ๆเช่น เราจะเข้าถึงตัวที่ 3 แถวที่ 6 ก็โดยแต่ละแถวมีความยาว 80 ตัว เราก็สามารถคำตำแหน่งได้คือ 80*6+3= 483 เป็นต้น

ข้อเสียเปรียบของโครงสร้างข้อมูลแบบนี้คือ การแทรกข้อมูลทำได้ยาก และขนาดของบรรทัดมีความยาวจำกัด

2.4.3 โครงสร้างแบบลิงค์ลีสต์ (linked list) ซึ่งจะเป็นโครงสร้างข้อมูลที่มีลักษณะเป็นโหนดประกอบด้วยข้อมูลและปอยน์เตอร์ สำหรับในกรณีนี้ต้องมี 4 ทิศทาง คือ ตัวซ้าย ตัวขวา ตัวบน ตัวล่าง

ข้อมูลได้เปรียบของโครงสร้างข้อมูลคือสามารถทำการแทรกคือลบข้อมูลได้รวดเร็ว เนื่องจากเพียงแค่เปลี่ยนปอยน์เตอร์ก็เป็นอันใช้ได้ สำหรับข้อเสียเปรียบก็คือกินเนื้อที่ในหน่วยความจำมากเป็นพิเศษ

2.4.4 โครงสร้างแบบลิงค์ลิสต์ของบรรทัด (linked list of line) โครงแบบนี้ดัดแปลงมาจากแบบที่สองกับแบบ ที่สาม จึงมีจุดเด่นที่สำหรับคือ มีความรวดเร็วในการแทรกบรรทัดเนื่องจากเป็นแบบลิสต์ เข้าถึงข้อมูลแต่ละตัวอักษรได้ง่าย และประหยัดหน่วยความจำกว่าแบบที่สามเมื่อแฟ้มมีขนาดใหญ่ การแทรกข้อมูลในบรรทัดรวดเร็จกว่าแบบที่สองพร้อมเคลื่อนย้ายข้อมูลเพียงบรรทัดเดียว

ข้อเสีย ก็มีเหมือนแบบที่สองคือ ความยาวบรรทัดจำกัด และมีการเสียพื้นที่ท้ายบรรทัดที่ว่าง

ในการพัฒนาเอดิเตอร์ตัวนี้เราเลือกใช้โครงสร้างข้อมูลแบบหลังสุด คือ ลิงค์ลิสต์ของบรรทัด เนื่องจากมีจุดเด่นดังที่กล่าวมา และจุดเสียเป็นข้อที่เรายอมรับได้ เช่น ความ

ยาวบรรทัดจำกัดนั้น เราก็ไม่ได้ต้องการจำนวนความยาวบรรทัดที่ยาวมากนัก

2.5 การจองเนื้อที่หน่วยความจำในเอ็มเอสดอส (Memory allocation in MS DOS)

เครื่องไอบีเอ็มที่ใช้เอ็มเอสดอสเป็นระบบจัดการนั้นสามารถที่จะจัดสรรเนื้อที่หน่วยความจำได้ขนาด 640 K ตั้งแต่ตำแหน่งที่ 00000 ไปจนถึง 09FFFFH หน่วยความจำที่มีตำแหน่งถัดจากนี้จะสงวนไว้เป็น รอมจัดการฮาร์ดแวร์ (BIOS rom) วีดิโอแรม(video refresh ram)

ภายใต้การควบคุมของดอสจะแบ่งเนื้อที่แรมเป็น 2 ส่วน คือ

2.5.1 เนื้อที่ส่วนระบบจัดการ (the operating system area)

2.5.2 เนื้อที่ส่วนโปรแกรมชั่วครู่ (the transient program area TPA)

เนื้อที่ส่วนระบบจัดการจะเริ่มตั้งแต่ตำแหน่ง 00000 ซึ่งจะมีส่วน ตารางการรองรับการขัดจังหวะ (interrupt vector table) เนื้อที่โปรแกรมของระบบจัดการรวมทั้งตารางและบัฟเฟอร์ต่าง ๆ

สำหรับเนื้อที่ส่วนโปรแกรมชั่วครู่(ต่อไปจะเรียกว่า ทีพีเอ) นั้น เป็นส่วนของหน่วยความจำถัดจากนั้นขึ้นมา ดอสจะมีกล่องควบคุมการจัดสรรเนื้อที่หน่วยความจำเหล่านี้ในแต่ละส่วนที่มีการจองเนื้อที่ ดอสมีฟังก์ชั่นการจัดสรรเนื้อที่อยู่ 3 อันคือ

ฟังก์ชั่น	การทำงาน
48h	จองเนื้อที่หน่วยความจำ (allocate memory block)
49h	ปลดเนื้อที่หน่วยความจำ (release memory block)
4Ah	ดัดแปลงเนื้อที่หน่วยความจำ (modify memory block)

2.6 การใช้งานฟังก์ชั่นการจัดสรรเนื้อที่

ในการใช้งานจำเป็นต้องเข้าใจการจัดสรรเนื้อที่หน่วยความจำในดอสเวลาเราโหลดโปรแกรมเข้ามารันเสียก่อน โปรแกรมในเอ็มเอสดอสที่ใช้รันได้จริงมีสองชนิดคือ โปรแกรมสกุลคอม(*.COM) และ สกุลอีเอ็กซ์อี (*.EXE) ซึ่งทั้งสองตัวมีการจองหน่วยความจำเริ่มต้นต่างกันคือ

2.6.1 สกุลคอม เมื่อดอสทำการโหลดโปรแกรมชนิดนี้ดอสจะทำการจองเนื้อที่หน่วยความจำไว้ตั้งแต่เริ่มส่วน ทีพีเอ ไปจนสุดหน่วยความจำ (rest of memory) ในการจะจองเนื้อที่หน่วยความจำเริ่มจึงทำไม่ได้ นอกจากจะมีการดัดแปลงขนาดของเนื้อที่ส่วนนี้เสียก่อน ดังนั้นในการจะจองเนื้อที่ในส่วนนี้เราต้อง ทำการเรียกฟังก์ชั่น 4Ah เสียก่อน จึงสามาถใช้ฟังชั่น 48h จองเนื้อที่ใหม่ได้

2.6.2 สกุลอีเอ็กซ์อี เมื่อดอสโหลดโปรแกรมเข้ามาดอสจองหน่วยความจำเท่ากับค่าสูงสุดที่จะต้องจองสำหรับโปรแกรม (MAX_ALLOC),(อยู่ในแฟ้มข้อมูลอยู่แล้ว รายละเอียดหาได้จาก หนังสือ แอดวานด์ เอ็มเอสดอส ของ เรย์),บวกกับจำนวนหน่วยความจำที่โปรแกรมต้องการเพิ่ม แต่ถ้าหน่วยความจำที่เหลืออยู่ไม่เพียงพอ ดอสก็จะอ่านค่าต่ำสุดที่โปรแกรมต้องการและทำการจองแทน ทั้งนี้ถ้าหน่วยความจำยังเหลือไม่พออีกดอสจะไม่ทำการรันโปรแกรมนั้น และแสดงข้อความบอกความผิดพลาดออกมา

ดังนั้นในการจองเนื้อที่หน่วยความจำในแฟ้มสกุลอีเอ็กซ์อี สามารถที่จะทำการจองได้เลยเพราะดอสจะจองเนื้อที่หน่วยความจำแค่จำเป็นเท่านั้น การใช้ในจะง่ายกว่าสกุลคอมเล็กน้อย

2.7 นำข้อมูลเข้าและข้อมูลออกจากลิสต์(insertion และ deletion)

2.7.1 การนำข้อมูลเข้าในลิสต์ จะมีทั้งหมดห้าขั้นตอนคือ

2.7.1.1 จองเนื้อที่หน่วยความจำ

2.7.1.2 ทำให้บรรทัดที่จะแทรกชี้ไปบรรทัดถัดไป

2.7.1.3 ทำให้บรรทัดก่อนหน้านี้ชี้ไปยังบรรทัดใหม่

2.7.1.4 ทำให้บรรทัดใหม่ชี้ไปยังบรรทัดก่อนหน้า

2.7.1.5 ทำให้บรรทัดหลังชี้ไปยังบรรทัดใหม่

2.7.2 การนำข้อมูลออกจากลิสต์ มีอยู่ 3 ขั้นตอน คือ

2.7.2.1 ทำให้บรรทัดก่อนหน้าบรรทัดที่จะลบชี้ไปบรรทัดถัดจากที่จะลบไป

2.7.2.1 ทำให้บรรทัดถัดจากบรรทัดที่จะลบชี้ไปบรรทัดก่อนหน้าบรรทัดจะลบไป

2.7.2.3 ปลดหน่วยความจำของบรรทัดนั้นออกไป

2.8 **ส่วนจัดการและแสดงผลเป็นภาษาไทย**

ส่วนนี้มีรายละเอียดปลีกย่อยมากและเป็นข้อมูลทางด้านเทคนิคจะขอกล่าวรายละเอียดในบทที่3

2.9 **การใช้งานร่วมและการขยายระบบ**

เนื่องจากจุดประสงค์ในการพัฒนาเอดิเตอร์นี้มีจุดประสงค์เพื่อนำไปร่วมและพัฒนาต่อให้เป็นระบบใหญ่ การเชื่อมโยงที่ง่ายและมีประสิทธิภาพย่อมเป็นสิ่งสำคัญ

การเชื่อมโยงที่ระบบนี้พัฒนาขึ้น เป็นการเชื่อมโยงภายใต้ระบบจัดการเอ็มเอสดอส โดยระบบจะสามารถเรียกใช้ซอฟท์แวร์ใด ๆ ก็ได้ ที่ทำงานในระบบเอ็มเอสดอส โดยเอดิเตอร์มีศักยภาพเป็นตัวโปรแกรมแม่ซึ่งสามารถผ่านตัวแปรผ่านไปยังตัวลูก โดยเอ็มเอสดอสสนับสนุนอยู่แล้ว สาม วิธี คือ

2.9.1 เอวิรอนเมนต์ (environment Block)

2.9.2 ส่วนท้ายของคำสั่ง (command tail)

2.9.3 เอฟซีบี 2 ตัว (two default fcb)

นอกจากนี้การส่งผ่านข้อมูลขนาดใหญ่ยังทำได้โดยส่งเป็นไฟล์ผ่านเอ็มเอสดอส

บทที่ 3

การคำนวณและการสร้าง

ตัวอักษรไทยที่ใช้งานปัจจุบัน

ตัวอักษรไทยที่ใช้กันในปัจจุบัน จะต้องได้รับการกำหนดรหัสขึ้นแทน รหัสที่ใช้แทนนี้หาก พิจารณาดูบนตาราง ISO646-1983 ซึ่งเป็นตาราง ASCII แล้วเราสามารถขยายเพิ่มต่อได้อีกเท่าตัว < 128 อักษร > แต่อย่างไรก็ตาม เครื่องคอมพิวเตอร์บางเครื่องรับรหัสเพียง 7 บิทเท่านั้น การขยายเพิ่มต่อจึงเสมือนการใช้หลักการ shift ซึ่งเมื่อเป็นเช่นนี้ รหัสภาษาไทยที่น่าจะใช้ได้ จึงต้องหลบรหัสควบคุมบางส่วนออกไป โดยเริ่มจากรหัส A0 เป็นต้นไป

ตารางของรหัส ISO646-1983 เป็นดังตารางที่ 1

					0	1	2	3	4	5	6	7
b4	b3	b2	b1									
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	¤	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	IS4	,	<	L	\	l	|
1	1	0	1	13	CR	IS3	-	=	M	]	m	}
1	1	1	0	14	SO	IS2	.	>	N	^	n	~
1	1	1	1	15	SI	IS1	/	?	O	_	o	DEL

ตารางที่ 1 รหัส ISO646 ซึ่งกำหนดรหัสไว้ครบแล้วใน 7 บิทแรก

สำหรับตัวอักษรไทยทั้งหมด หากจะพิจารณาดูจะประกอบด้วยตัวอักษรที่สำคัญดังนี้

พยัญชนะ ได้แก่

ก ข ค ฆ ง จ ฉ ช ซ ฌ ญ ฎ ฏ ฐ ฒ ด ต ถ ท ธ น บ ป ผ ฝ พ ฟ ภ ม ย ร ฤ ล ว ศ ษ ส ห ฬ อ ฮ < พยัญชนะมีทั้งหมด 44 ตัวกับ ตัวอักษรที่ปัจจุบันได้เลิกใช้แล้ว

สระได้แก่ ะ ั า ำ ิ ี ึ ื เ แ โ ใ ไ ุ ู ๅ และ พินทุ รวม 18 ตัว

วรรณยุกต์และการันต์ ได้แก่ ่ ้ ๊ ๋ ์ รวม 5 ตัว

เครื่องหมายพิเศษที่ใช้งานที่ใช้ทั่วไป ๆ ฯ

เครื่องหมายพิเศษเฉพาะ ได้แก่ ฟองมัน โคมูตร ยามักการ เครื่องหมายเงินบาท รวมเครื่องหมายพิเศษ 7 ตัว

เลขไทย ทั้งหมด 10 ตัว

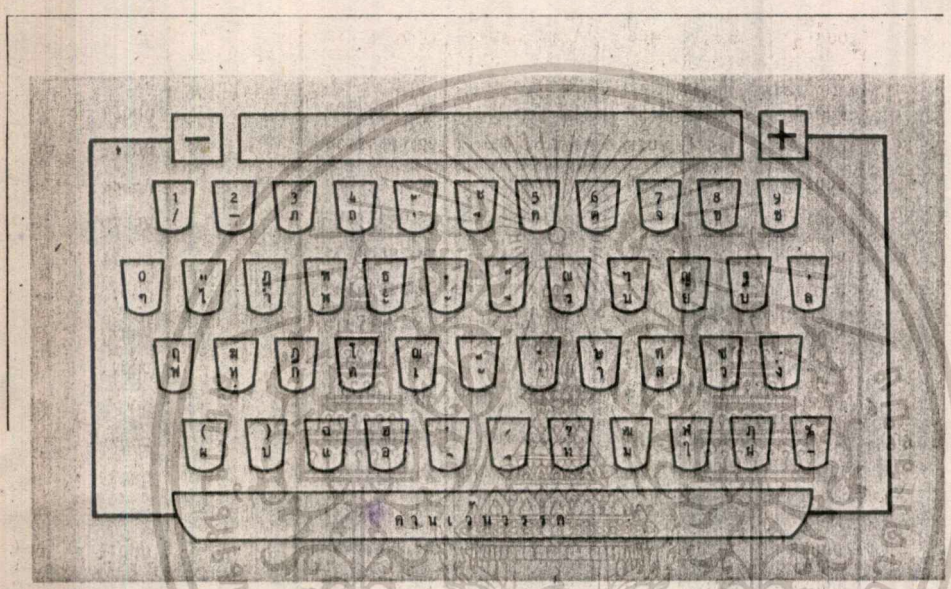
ดังนั้นตัวอักษรทั้งหมดที่ใช้มีจำนวน 46+18+5+7+10 รวม 86 ตัวอักษร

ตัวอักษรไทยกับคีย์บอร์ด

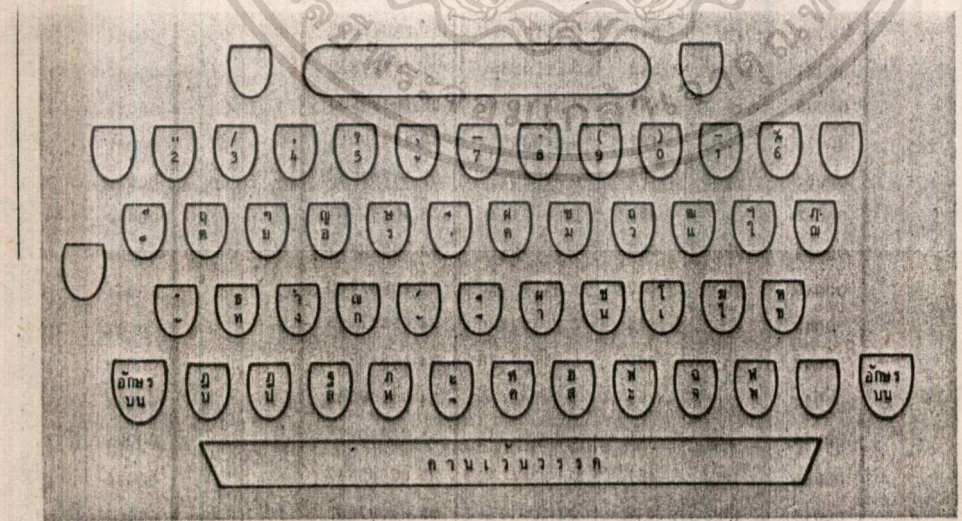
คียบอร์ดของคอมพิวเตอร์ก็ใช้การจัดวางเหมือนกับพิมพ์ดีดไทยทั่งไป การจัดวางคีย์บอร์ดของเครื่องพิมพ์ดีดไทยมีสองรูปแบบคือ รูปแบบดั้งเดิมมา และแบบที่นักวิจัยชื่อ สฤษดิ์ ปัตตะดชติได้วิจัยและเสนดแนะ โดยรัฐบาลก็ยอมรับและเคยประกาศให้หน่วยงานราชการใช้แต่ความคุ้นเคยของผู้ใช้ก็ยากที่จะแก้ไขและเปลี่ยนแปลงได้ จนปัจจุบันการจัดคีย์แบบเดิมมีผู้

ใช้มากกว่า 90% ของจำนวนผู้ใช้ และเคยมีผู้สำรวจคีย์บอร์ดแบบ ปัตตะโชติเพียงประมาณ 8% ของทั้งหมดและมีแนวโน้มไม่เพิ่มขึ้น เพราะแม้แต่แบบเรียนของกระทรวงศึกษาธิการก็กำหนดแป้นพิมพ์แบบเดิม

การจัดวางตำแหน่งคีย์บอร์ดทั้งสองได้แสดงดังรูป



รูปที่ 3 ตำแหน่งการจัดแป้นพิมพ์ตามแบบเรียนสอนพิมพ์ดีดไทย พช 121



รูปที่ 4 ตำแหน่งการจัดแป้นพิมพ์แบบปัตตะโชติ

สำหรับการจัดแป้นพิมพ์ทางด้านไมโครคอมพิวเตอร์หรือเทอร์มินอร์นั้น มักจะอยู่ภายใต้

การควบคุมของซอฟท์แวร์ ดังนั้นจึงแก้ไขเปลี่ยนแปลงให้เป็นอะไรก็ได้ โดยเฉพาะอย่างยิ่ง ผู้เขียนซอฟท์แวร์ จะเขียนด้วยภาษาแอสแซมบี้ร่วมกันอินเตอร์รัพคีย์บอร์ด แล้วแก้ไขสร้างตารางการปรับคีย์ เพื่อสร้างคีย์บอร์ดเป็นอะไรก็ได้

ข้อพิจารณาทางภาษาศาสตร์กับการพัฒนา

โครงสร้างการพัฒนาการทางด้านซอฟท์แวร์ จำต้องประกอบด้วยหลักวิชาการและอาศัยประสบการณ์ประกอบร่วมกัน จึงจะทำให้ซอฟท์แวร์นี้ฯมีประสิทธิภาพยิ่ง โครงสร้างของภาษาไทยเป็นสิ่งที่เป็นธรรมชาติ จนยากที่จะหากฎเกณฑ์ตายตัวได้ ตัวอักษรดังกล่าวนี้ จึงกล่าวได้ว่าเป็นตัวอักษรสำหรับแลกเปลี่ยนข้อมูลภาษาไทย แต่อย่างไรก็ตาม ตัวอักษรหลายตัวอาจมีผู้แย้งว่า เลิกใช้กันมานานแล้วจะเก็บไว้ทำไม คำตอบสำหรับกรณีนี้เห็นจะได้ว่า มีไว้ไม่เสียหาย

ปัญหามีอยู่ว่า ในการใช้งานไมโครคอมพิวเตอร์ทั่วไป จำต้องใช้ระหัสแสดงผล และเทคโนโลยีปัจจุบันยังไม่เอื้ออำนวยต่อการแสดลผลแบบกราฟฟิกที่ใช้ระหัสเดี่ยวได้ การใช้รหัสผสมทั้งบนจอภาพและเครื่องพิมพ์ถึง 4 ระดับมาเป็น 3 ระดับ

ตัวอักษรผสมที่พิจารณานำมาใช้ควรประกอบด้วย

่ ้ ๊ ๋ ั ิ ี ึ ื ็ ์ ๎ ๐ ๑ ๒ ๓ ๔ ๕ ๖ ๗ ๘ ๙

รหัสผสมทั้งหมดนี้มี 25 ตัว แต่ในการแสดงผลเฉพาะบางอย่าง อาจนำตัวอักขระบางตัวกระจายออก เพื่อให้แสดงผลได้สวยงามขึ้น แต่ก็ต้องเปลืองรหัสมากขึ้น เช่น

พ รวมกับ ' เพื่อให้ได้ ฟ ที่หางยาวขึ้น

เ รวมกับ เพื่อให้ได้ ไ มีส่วนสูงมากขึ้น

(ที่สำคัญคือ ส่วนล่างของ เ จะใช้ร่วมกับ สระ เ ไม่ได้เพราะส่วนจุดการแสดงผลจะขาด

ไป ทำให้ไม่สวยงาม จึงต้องแยกรหัส ไ ออกเป็น 2 ตัว)

การกำหนดรหัสลงในตารางจึงควรยึดหลักการที่สำคัญคือ รหัสที่กำหนดเมื่อใช้กับเครื่องที่รับ 7 บิท แล้วใช้หลักการ shift จะมีผลกระทบกระเทือนน้อยที่สุด นอกจากนี้ รหัสที่กำหนดนี้ จะกระทบกระเทือนต่อรหัสควบคุมเฉพาะบางอย่างของเครื่องคอมพิวเตอร์น้อยที่สุด เช่น กระทบกับโปรแกรมจัดระบบงาน กระทบกับโปรแกรมประยุกต์อื่นๆ

สระผสมมีความสำคัญอย่างไร

สระผสมในที่นี้หมายถึง การนำรูปสระสองตัวมารวมกันเพื่อกำหนดเป็นระหัสหนึ่งตัว เช่น ี รหัสผสมมักใช้สำหรับการแสดงผลด้วยหลักภาษาไทย 3 บรรทัด ซึ่งต้องการรหัสเพิ่มขึ้นอย่างน้อย 25 ตัว (หรือมากกว่า) ความสำคัญของรหัสผสมจึงเน้นเฉพาะการใช้งานเรื่องการแสดงผล เช่น แสดงผลบนจอภาพ แสดงผลบนเครื่องพิมพ์ รูปแบบผสมนี้ จะช่วยทำให้การจัดแสดงบนภาคแสดงดีขึ้นและสะดวกขึ้น เช่น การพิมพ์ตัวอักษรของเครื่องพิมพ์ดอทแมททริกซ์กับงานแลกเปลี่ยนข้อมูลและข่าวสาร ไมโครคอมพิวเตอร์เครื่องหนึ่งอาจแสดงผลด้วยอักษะผสมได้ แต่เก็บข้อมูลในหน่วยความจำแบบแยก เช่น ิ ' เป็น 2 รหัส

การขยายรหัสภาษาสำหรับอักษรภาษาไทย

ในการแสดงภาษาไทยนั้นมี ตัวอักษรที่จำกัดอยู่เพียง 86 ตัวดังที่กล่าวมาแล้ว แต่เราอาจต้องการรหัสที่เป็นการผสมตัวอักษรอีก 25 ตัว ซึ่งก็ยังบรรจุลงในตารางขนาด 128 ช่องได้

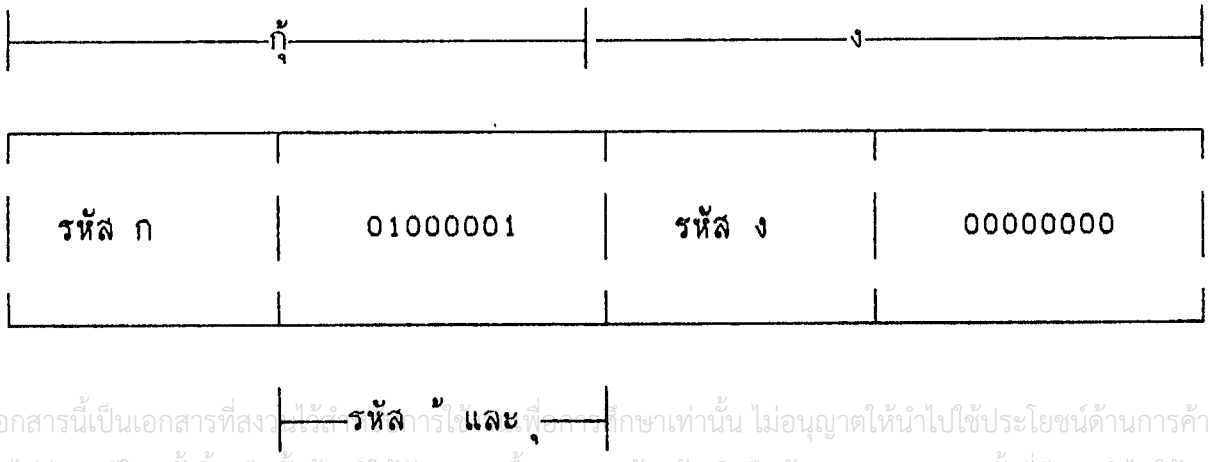
ภาษาไทยเป็นโครงสร้างที่จัดตัวอักษรเป็น 4 ระดับ ซึ่งการประมวลผลต่างๆ จะต้องมีซอฟท์แวร์จัดการอยู่ด้วยเสมอ ทำให้ซอฟท์แวร์จากต่างประเทศมักมีปัญหาในการทำ

งานกับข้อมูลภาษาไทยในระบบจัดการระดับโดยตรง ทั้งนี้เพราะตำแหน่งทางพิสิศัลที่มองเห็น กับตำแหน่งทางลอจิคอลที่อักษรอยู่ในหน่วยความจำมักไม่ตรงกัน การกำหนดตำแหน่งการกำหนดรูปแบบการประมวลผลจึงมีปัญหามากเช่น ก ุ ้ ง และ ก ้ ุ ง ในทางลอจิคอลแล้วต่างกัน แต่ทางพิสิคัลแล้วเหมือนกัน เป็นต้น

ผู้เขียนได้ลองขยายรหัสพิสิคัลกับลอจิคอลให้ตรงกัน แล้วจัดการกับระบบภาษาไทย 25 บรรทัด โดยยึดหลักการง่ายๆ ดังนี้

โครงสร้างรหัสสองไบท์

ลักษณะการกำหนดรหัสนี้ จึงเหมือนกับ การแสดงผลใน ไอบีเอ็มพีซี อยู่แล้วคือ ไบท์แรกเป็นระหัสแอสกี้ ไบท์ที่สองคือ แอตตริบิ้วโดยใช้ไบท์ที่สองนี้ขยายรหัสออกอีก จึงเท่ากับว่าเราสามารถกำหนดรหัสภาษาไทยได้ ถึง 65536 ตัว โดยมีหลักการกำหนดดังรูปที่ 1



การแทนข้อมูลด้วยรหัสสองไบท์

ด้วยวิธีการนี้เราสามารถสร้างรหัสได้ครบทุกตัว สำหรับการแสดงผลภาษาไทย และหากใช้เทคนิคการลดคำโดยใช้คำที่เป็นไปได้และมีความมหมายเท่านั้น เช่น กิ้ คือ แสดงทั้งสระล่างและสระบนพร้อมกันไม่มีเราจะลดจำนวนรหัสลงได้มากจากการทดลองวิจัยที่ผ่านมา เราจะพบว่าจะใช้ระหัสเพียงประมาณ 2800-3200 ตัว ซึ่งอยู่ในวิสัยที่จะใช้รอม ในการกำเนิดตัวอักษร และที่สำคัญ วิธีการนี้ จะทำให้เครื่องพิมพ์ที่มีความละเอียดสูงสามารถที่พิมพ์ภาษาไทยได้เร็วขึ้น คือพิมพ์ภาษาไทยครั้งเดียวได้ทุกระดับได้โดยตรง

หลักการเขียนโปรแกรมจัดระดับภาษาไทย

3.1 การทดลอง

จากที่ได้บรรยายมาข้างต้นนั้นแล้วว่า ในการแสดงออกจอ ของภาษาไทยและอังกฤษ นั้น ไม่เหมือนกัน คือ ในระบบภาษาไทยนั้นมีทั้งวรรณยุกต์ และพยัชนะ ที่อยู่ที่ระดับบนและล่างดังนั้น ในการเขียนโปรแกรมจัดระดับภาษาไทยเราต้องมีการเช็คว่าตัวอักษรที่รับเข้าจากคีย์บอร์ด โดยการรับคีย์ เราใช้อินเตอร์รัพ ของไบออส ดังนี้

```
MOV AH , 0
INT 16H
```

ซึ่งจะได้ตัวอักษรอยู่ในลักษณะรหัสแอสกี้ อยู่ในรีจิสเตอร์ AL เราจะเช็คว่าในขณะนั้นอยู่ใน โหมด ภาษาไทย หรือ โหมดภาษาอังกฤษ ดังตัวอย่างโปรแกรมนี้

```
        cmp mode,1              ;1 is thai mode  check mode
        jne is_eng_mode
        jmp is_thai_mode        ;is thai mode then goto is_thai_mode
is_eng_mode:
        call print_english
        ret
is_thai_mode:
        call print_thai
```

ในกรณีที่อยู่ในโหมดภาษาอังกฤษ เราไม่จำเป็นที่ต้องไปเปิดตารางเพื่อเปลี่ยนระหัสให้เป็นระหัสภาษาอังกฤษ แต่ในกรณีที่อยู่ในโหมดภาษาไทย เราต้องทำการเปิดตารางเพื่อเปลี่ยนรหัสเป็นภาษาไทย และต่อจากนั้นเราจะเช็คว่าเลขระหัส อยู่ในระดับกลาง หรือ ล่าง หรือ บน โดยใช้การเปรียบเทียบเลขละหัสแอสกี้ตามตาราง

ซึ่งจากตารางเราจะเห็นว่า รหัสภาษาไทยจะเริ่มต้นแต่รหัส 161d(a1h) จะเป็นตัวอักษร กและตัวอักษรตัวสุดท้าย อ ตรงกับรหัส 203d(cb h) ซึ่งตัวอักษรเหล่านี้จะอยู่ในระดับกลาง ส่วนตัวอักษรระดับล่างคือ สระ ุ ู ตรงกับรหัส 215 และ 216 ส่วนตัวอักษรตัวบนจะเริ่มต้นตั้งแต่ 217 ถึง 228 ส่วนระหัสตั้งแต่ 228 เป็นรหัสผสม

ดั้งนั้นในการเขียนโปรแกรมนั้นเราต้องใช้คำสั่งเช็คว่าตัวอักษรที่เรารับเข้ามานั้นเป็นตัวอักษรในระดับใดดั้งนั้น

```
        cmp al,215              ;is in lower position then check char
        je is_215_216           ;check is lower character
```

```
        cmp al,216
        je is_215_216
```

ส่วนนี้เป็นโปรแกรมที่ใช้เช็คว่าเป็นตัวอักษรระดับล่าง

```
        cmp al,217              ;check is upper character
        jnb is_upper_char
        ret
is_upper_char:                  ;subroutine to write upper characte
```

ส่วนนี้เป็นโปรแกรมที่เราใช้เช็คว่าเป็นรหัสบน

ซึ่งจากโปรแกรมนั้นเราจะได้ว่ากรณีที่ตัวอักษรที่ไม่ใช่บนและล่างก็จะเป็นตัวอักษรระดับกลาง

ในส่วนของโปรแกรมนั้นจะมีปัญหาตรงส่วนระดับบนเช่นกรณีที่ เราพิมพ์ ^ แล้วพิมพ์ ตัวอักษรที่ได้จะเป็น ี่ ดังนั้นเราจะต้องมีการเช็คว่าตัวอักษรตัวก่อนหน้านั้นเป็นตัวอักษรใด ถ้าตัวอักษรก่อนหน้านี้เป็นตัวบน และตัวอักษรที่กดตามมาเป็นตัวอักษรตัวบนอีกเหมือนกัน เราจะต้องทำงานเปิดตารางเพื่อเปรียบเทียบหาสระผสม ตัวตัวอย่างโปรแกรมดังต่อไปนี้

```
        cmp bl,222      ;bl have range 217-222
        jbe is_looktable1
        jmp is_looktable2
is_looktable1:
        cmp al,228
```

```
            jne is_not_2178
            cmp bl,217
            jne is_not_2178
            mov al,242
            jmp upper_con1
is_not_2178:
            cmp al,224  ;238,243,247,251,234,230
            jne is_not_224
            mov bh,0
            sub bl,217
            mov al,table1[bx]
            jmp upper_con1
is_not_224:
            cmp al,225      ;239,244,248,252,235,231
            jne is_not_225
            mov bh,0
            sub bl,217
            mov al,table2[bx]
            jmp upper_con1
is_not_225:
```

```
            cmp al,226          ;240,245,249,253,236,232
            jne is_not_226
            sub bl,217
            mov bh,0
            mov al,table3[bx]
            jmp upper_con1
is_not_226:
           cmp al,227           ;241,246,250,254,237,233
           jne is_not_227
           sub bl,217
            mov bh,0
            mov al,table4[bx]
           jmp upper_con1
is_not_227:
           jmp upper_con1
is_looktable2:
           cmp bl,230
           jae is_more_230
           jmp upper_con1
is_more_230:
```

```
            cmp al,223
            ja is_more_223
            jmp upper_con1
is_more_223:
            cmp al,228
            jne is_less_228
            cmp bl,239
            jb is239_242
            cmp bl,242
            ja is239_242
            mov al,242
is239_242:
            jmp upper_con1
is_less_228:
            cmp bl,233    ; 230,231,232,233
            ja is_more_233
            mov ah,0
            sub ax,224
            mov bx,ax
            mov al,table5[bx]
```

```
            jmp upper_con1

is_more_233:

            cmp bl,237    ; 230,231,232,233

            ja is_more_237

            mov ah,0

            sub ax,224

            mov bx,ax

            mov al,table6[bx]

            jmp upper_con1

is_more_237:

            cmp bl,242    ; 230,231,232,233

            ja is_more_242

            mov ah,0

            sub ax,224

            mov bx,ax

            mov al,table7[bx]

            jmp upper_con1

is_more_242:

            cmp bl,246    ; 230,231,232,233

            ja is_more_246
```

```
            mov ah,0
            sub ax,224
            mov bx,ax
             mov al,table8[bx]
            jmp upper_con1
is_more_246:
            cmp bl,250
            ja is_more_250
            mov ah,0
            sub ax,224
            mov bx,ax
            mov al,table9[bx]
            jmp upper_con1
is_more_250:
             sub al,224
             mov ah,0
             mov bx,ax
             mov al,table10[bx]
```

3.2 ข้อมูลที่ได้จากการทดลอง

ผลจากการทดลองเรา จะพบว่าเมื่อเราใช้คำสั่ง cmp เพื่อเปรียบเทียบค่าของ

รหัส แล้วหลังจากนั้นจะใช้คำสั่ง jmp เพื่อให้มีการกระโดดไปทำงานตามส่วนของโปรแกรม เราจะพบว่า เมื่อเราทำการเปรียบเทียบเนื่องจาก การเก็บข้อมูลในหน่วยความจำของคอมพิวเตอร์ บิทที่ 0 ถึง 6 ใช้เก็บค่า ส่วนบิทที่ 7 เราใช้แทนเครื่องหมายว่าเป็นบวก หรือ ลบ ดั้งนั้นในการเลือกคำสั่ง jmp ที่เหมาะสม เพราะมิฉนั้นจะทำให้เกิดการเปรียบเทียบค่าที่ถูกต้อง

รูปแบบของคำสั่ง jmp มีดังนี้

jmp [illegible]

บทที่ 4

การทดลองและผลการทดลอง

4.1 การทดลองการจัดระดับ

ในการจัดระดับภาษาไทยนั้น เราต้องมีการตรวจสอบว่าตัวอักษรที่รับเข้ามา เป็นตัวอักษรในระดับใด เช่น ระดับบน กลาง หรือ ล่าง โดยเราต้องใช้คำสั่งตรวจสอบระหัสที่รับเข้ามา ว่าจะอยู่ในระดับใด โดยตัวอักษรระดับกลางนั้นจะมีระหัสแอสกี้ 161-214 ระดับล่าง มีระหัส ตั้งแต่ 215-216 ระดับบนคือ 217-254 ซึ่งในการตวรจสอบระหัสนั้นเราใช้คำสั่ง CMP เพื่อเปรียบเทียบระหัสว่าอยู่ในระดับใด ดังตัวอย่าง

4.1.1 การทดลองการตรวจสอบว่าเป็นระดับบนหรือไม่

```
cmp linestatus,2        ;check is upper character
jne is_in_middle
cmp al,217              ;check is upper character
jnb is_upper_char
ret
```

4.1.2 การทดลองการตรวจสอบว่าเป็นระดับล่างหรือไม่

```
cmp linestatus,0        ;1 is middle , 0 is lower and 2 is up
jne not_inlower_status
cmp al,215              ;is in lower position then check char
je is_215_216           ;check is lower character
cmp al,216
je is_215_216           ;check is lower character
```

4.2 ผลการทดลอง

สามารถที่จะตรวจสอบและจัดระดับการแสดงภาษาไทยได้อย่างถูกต้อง ปัญหารที่เกิดขึ้นเล็กน้อยก็คือการรวมตัวอักษรบนเพื่อการแสดงผล เช่น การรวมสระอิ กับ ไม้เอก (ซึ่งระหัสแอสกี้เป็น 217 224 ตามลำดับ) ให้เป็นสระรวมสระอิกับไม้เอก ซึ่งมีระหัสแอสกี้เป็น 238 การแก้ปัญหาทำได้โดยการนำไปเปิดตารางอีกครั้งหนึ่ง

4.3 การเรียกชั้นของระบบจัดการ (OS SHELL)

การเรียกชั้นของระบบจัดการทำได้โดยการใช้ฟังก์ชั่น 4BH ของดอสทำการรันโปรแกรมคอมมานด์โปรแซสเซอร์ (โดยทั่วไปจะเป็น command.com) ซึ่งการทำเช่นนี้ทำให้เราสามารถที่ใช้คำสั่งต่าง ๆในดอสมาใช้ในเอดิเตอร์ของเราได้ด้วย

การเรียกใช้คอมมานด์โปรเซสเซอร์ทำได้ตามโปรแกรมตัวอย่างข้างล่าง ซึ่งเป็นโปรแกรมที่เขียนขึ้นเพื่อศึกษาการใช้งานนี้โดยเฉพาะ แล้โปรแกรมนี้ถูกพัฒนาเป็นส่วนหนึ่งของเอดิเตอร์ที่สร้างขึ้น

```
code segment
          assume cs:code
          org 0
code_begin equ $
          org 100h
start:
          jmp skip
cmd_tail db ' /c dir/w ',0dh
stackptr dw ?
stackSEg dw ?
comma db 'command.com',0
;========Parameter Block========
parameter dw 0
          dw offset cmd_tail
c1        dw 0 ; segment cmd_tail
          dd -1 ;fcb1
          dd -1 ; fcb2
;================================
skip:    ; modify allocate memory
          mov ah,4ah
          mov bx,(offset code_end -offset code_begin+15)shr 4
          int 21h
          jnc cont
          jmp exit
cont:     mov c1,cs
          mov stackptr,sp
          mov stackseg,ss
          mov ah,4bh
          mov al,0
          mov dx,offset comma
          mov bx,offset parameter
          int 21h
```

```
            mov sp ,cs:stackptr
            mov ss ,cs:stackseg
exit:     mov ah,4ch
            int 21h
code_end:
code ends
end start
```

ปัญหาที่สำคัญในการเรียกคอมมานด์โปรเซสเซอร์ก็คือดอสจะไปอ่านคอมมานด์โปรเซสเซอร์จากไดว์ เอ เท่านั้น และถ้าหากว่าในไดว์ที่กำลังใช้งานอยู่(default disk drive)มีโปรแกรมคอมมานด์โปรเซสเหมือนกัน เครื่องจะอ่านจากไดว์ที่ใช้งานอยู่จากนั้นจึงไปอ่านในไดว์ เอ อีกครั้ง ปัญหาที่เกิดขึ้นก็คือเมื่อคอมมานด์โปรเซสเซอร์ที่ไดว์ทั้งสองเป็นคนระตัวหรือแม้แต่คนละรุ่นกัน(version) เครื่องจะมีแฮงค์ ไม่สามารถใช้งานต่อไปได้ ต้องทำการรีเซทเครื่องใหม่จึงจะใช้งานเครื่องอีกได้

4.4 แนวทางแก้ปัญหาที่เกิดขึ้น

ปัญหาที่เกิดขึ้นเป็นปัญหาใหญ่ เพราะเครื่องถึงกับหยุดทำงานจะต้องแก้ไขให้ได้ ไม่เช่นนั้นเราจะไม่สามารถมั่นใจในข้อมูลที่เราป้อนเข้าไป ซึ่งจะสูญหายไปหมด เมื่อเครื่องหยุดทำงาน จากการใช้ความพยายามในการทดลอง การแก้ปัญหาดังกล่าวมีแนวทางแก้ไขก็โดยการบอกดอสให้อ่านคอมมานด์โปรเซอร์ที่ไดว์ที่ใช้งานอยู่เท่านั้น ซึ่งถ้าในไดว์นี้ไม่มีตัวนี้อยู่เครื่องก็จะเพียงขึ้นข้อความแสดงว่าหาคอมมานด์โปรเซอร์ไม่พบเท่านั้น ไม่ถึงกับหยุดทำงาน การมองหาทางออกต่อไปคือทำอย่างใดดอสถึงจะรู้ได้

4.5 การบอกดอสให้รู้ว่าให้อ่านคอมมานด์โปรเซสเซอร์จากที่ไหน

จากคู่มือดอสเวอร์ชั่นสาม การกระทำดังกล่าวทำได้ง่าย ๆแค่เพียงใช้คำสั่ง SET ดังตัวอย่างข้างล่าง

```
set comspec c:command.com
```

แต่เนื่องคำสั่งดังกล่าวไม่สามารถที่จะใช้ได้ในดอสเวอร์ชั่น 2 ระหว่างการค้นคว้าพบหนังสือเล่มหนึ่ง อ้างว่าโปรแกรมที่อยู่ในหนังสือดังกล่าวสามารถแก้ปัญหาที่ว่านี้ได้ โปรแกรมดังกล่าวอยู่ คือ

```
dsk equ 939h
code segment
assume cs:code,ds:code
    org 5ch
```

```
param label byte
      org 100h
start:
      mov ah,30h
      int 21h
      cmp al,2
      jnz bad_ver
;find alloation of a:\command.com
      mov ax,3522h
      int 21h
      mov al,'c'
;set up drive c:for default
      mov ah,[param]
      or ah,ah
      jz no_para
      mov al,ah
      add al,'a'-1
no_para:
      mov es:[dsk],al
      mov [new_d],al
      mov dx,offset mes_d
      mov ah,9
      int 21h
      int 20h
bad_ver:
      mov dx,offset bad_ver_mess
      mov ah,9
      int 21h
      int 20h
bad_ver_mess db 'DOS must be 2 $ '
mes_d db  ' COMMAND.COM now can read from drive '
new_d db  'C$'
code ends
```

```
end start
```

แต่จากการทดลองทางจริงปรากฏว่า โปรแกรมดังกล่าวไม่สามารถที่จะแก้ไขปัญหานี้ได้แต่อย่างใด การหาทางออกจึงต้องทำต่อไป จากคู่มือดอสเล่มเดียวกันในหัวข้อคำสั่ง set มีการกล่าวถึงคำว่า เอนวิรอนเมนต์ (environment) ทั้งยังบอกด้วยว่าเมื่อเราใช้คำสั่ง set แล้ว ตัวแปรที่ตั้งไว้ ไปแก้ที่ส่วนดังกล่าว แนวทางการแก้ไขจึงมุ่งไปที่ส่วนดังกล่าว

ในคู่มือ ดอสเทคนิดเคิล (DOS technical reference)กล่าวว่า เอนวิรอนเมนต์ที่ว่าจะถูกลอกเลียน (copy)ไปยังโปรแกรมที่กำลังรันอยู่ด้วย การเข้าถึงมันสามารถทำได้โดยการอาจค่าเซกเมนต์(segment) จากส่วนโปรแกรมเซกเมนต์พรีฟิกซ์ ตำแหน่งที่ 2CH จึงลองใช้โปรแกรมดีบัคดูที่ตำแหน่ง พีเอสพีดังกล่าว ดังรูป

```
C>symdeb
Microsoft (R) Symbolic Debug Utility  Version 4.00
Copyright (C) Microsoft Corp 1984, 1985.  All rights reserved.

Processor is [8086]
-d0 4f
82FF:0000  CD 20 00 A0 00 9A EE FE-1D F0 8E 09 04 7A 2B 0A
M . ..n~.p...z+.
82FF:0010  04 7A 56 09 04 7A 6F 5F-01 01 01 00 02 FF FF FF
.zV..zo_........
82FF:0020  FF FF FF FF FF FF FF FF-FF FF FF FF F1 79 46 01
.............qyF.
82FF:0030  6F 5F 00 00 00 00 00 00-00 00 00 00 00 00 00 00
o_..............
82FF:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
................
```

จะเห็นได้ว่าตำแหน่ง 2Ch มีค่าเป็น 4679 นำตัวเลขนั้มเป็นเซกเมนต์ อ้างออฟเซทเท่ากับศูนย์ ดังรูป

```
-d79f1:0  4f
79F1:0000  50 41 54 48 3D 00 43 4F-4D 53 50 45 43 3D 41 3A
PATH=.COMSPEC=A:
```

```
79F1:0010  5C 43 4F 4D 4D 41 4E 44-2E 43 4F 4D 00 00 88 07
\COMMAND.COM....
79F1:0020  4D F4 79 0A 09 F6 2B FF-E8 D7 43 B4 4A BB 00 07
Mty..v+.hWC4J;..
79F1:0030  CD 20 00 A0 00 9A F0 FF-0D F0 8C 02 6F 5F 99 02
M . ..p..p..o_..
79F1:0040  6F 5F E2 04 6F 5F 6F 5F-01 01 01 00 02 FF FF FF
o_b.o_o_........
```

จะเห็นได้ว่าที่ตำแหน่ง 1a เป็นต้องไปมีจุดน่าสนใจคือมีชื่อของ a:command.com อยู่ทด ลองโดยเปลี่ยนจากตัวเอ เป็นตัว ซีจะได้

```
-e79f1:0e 43
-d79f1:0 4f
79F1:0000  50 41 54 48 3D 00 43 4F-4D 53 50 45 43 3D 43 3A
PATH=.COMSPEC=C:
79F1:0010  5C 43 4F 4D 4D 41 4E 44-2E 43 4F 4D 00 00 88 07
\COMMAND.COM....
79F1:0020  4D F4 79 0A 09 F6 2B FF-E8 D7 43 B4 4A BB 00 07
Mty..v+.hWC4J;..
79F1:0030  CD 20 00 A0 00 9A F0 FF-0D F0 8C 02 6F 5F 99 02
M . ..p..p..o_..
79F1:0040  6F 5F E2 04 6F 5F 6F 5F-01 01 01 00 02 FF FF FF
o_b.o_o_........
-nt.com
-l
-g

 Volume in drive C is VDISK  V3.2
 Directory of  C:\

MASM       EXE     LINK       EXE     EXE2BIN  EXE     H          BAT
SYMDEB     EXE
SEE        EXE     PAU        COM     D        COM     DOSVAR     COM
COMMAND    COM
```

```
T          ASM     T          COM     P                    RWMAIN    COM
Q
        15 File(s)     113920 bytes free

Program terminated normally (2)
```

จากการสังเกตุจะเห็นได้ว่าดอสไปอ่านคอมมานด์โปรเซสเซอร์ที่ไดว์ ซี จริง การแก้ไขในการที่จะทำให้ดอสไปอ่านที่ไดว์ซีในโปรแกรมจิง อาจแก้ไขได้โดยเติมคำสั่งต่อไปนี้

```
code segment
     org 2ch
envi label word
     org 100h
start:
          .
          .
          .
          mov ax,cs[envi]
          mov es,ax
          mov es:[0e],43h
          .
          .

code ends
end start
```

เมื่อทดลองแก้ปัญหาโดยวิธีดังกล่าวจะเห็นได้ว่าปัญหาหมดไปโดยสิ้นเชิง แต่ก็ต้องใช้ความพยายามและการค้นคว้าอยู่พอสมควรทีเดียว แต่ก็มีข้อที่ต้องคำนึงในเวลาใช้อีกคือ เนื่องจากว่าการบูทด้วยดอสเวอร์ชั่นสูงย่อมสามารถที่จะคอมมานด์โปรเซสเซอร์ของเวอร์ชั่นต่ำกว่าได้ แต่ถ้าเราบูทด้วยดอสต่ำย่อมไม่สามารถที่จะใช้คอมมานด์โปรเซสเซอร์ของดอสเวอร์ชั่นสูงได้ แต่ผลที่ได้ก็เพียงแค่เครื่องแสดงข้อความว่า ดอสผิดรุ่นเท่านั้น(Incorrect DOS version)

บทที่ 5

สรุปและวิจารณ์

ระบบที่พัฒนาขึ้นยังไม่สมบูรณ์ แต่ก็สามารถบรรลุจุดประสงค์ที่ต้องการได้ คือ สามารถแสดงผลเป็นภาษาไทย 3 ระดับได้ และสามารถเชื่อมโยงกับระบบอื่น ๆที่จะพัฒนาขึ้นโดยไม่อยากนักเนื่องจาก สามารถที่จะใช้ชั้นของระบบจัดการและการส่งค่าตัวแปรตั้งแต่ 2-3 ไบท์ผ่านในเอนวิรอนเมนต์จนถึงขนาดใหญ่ ๆเป็นหลาย ๆกิโลไบท์ผ่านแฟ้มข้อมูลในดิสต์ แต่ข้อบกพร่องที่ยังไม่พบ(run time error)ก็อาจมีบ้าง เนื่องจากเวลาในการตรวจสอบมีจำกัด ต้องขออภัยไว้ใน ณ ที่นี้ด้วย อนึ่งการแก้ไขจุดบกพร่องแต่ละจุดทำได้ยากลำบากมาก เนื่องจากหนังสือหรือเอกสารต่าง ๆในเรื่องนี้มีน้อยมาก หรือมีบางเล่มที่พบก็ไม่สามารถแก้ไขได้จริง เป็นเพียงโปรแกรมในกระดาษเท่านั้น การแก้ไขที่ทำได้ก็โดยการทำการทดลองซ้ำหลาย ๆ ครั้ง ซึ่งเป็นการสิ้นเปลืองเวลาอย่างยิ่ง งานที่ได้จึงไม่มากเท่าที่ควร จึงขอฝากเรื่องนี้ไว้กับผู้ที่อยู่ในวงการหนังสือและวงการศึกษาช่วยตระหนักในเรื่องนี้ด้วย

ภาคผนวก

ส่วนข้อมูล. 1-4.

การจัดหน่วยความจำ 4-11

ส่วนขัดการจอภาพ. 12-13.

ส่วนอินเตอร์เฟซเป็นมอนิเตอร์. 14-17

ส่วนจัดการหน่วยความจำแบบเพจ. 17-20

ส่วนขัดการขั้นตอนจัดใหม่. 20-27

ส่วนขัดการขึ้นและลงของตัวชี้. (UP-DOWN CURSOR) 27-31

ส่วนขัดการเคลื่อน(เลื่อน)ข้อมูล. 32-36

ส่วนขัดระดับล่างภาษาไทย 36-53

ส่วนควบคุมการป้อนข้อมูล. 54-56

โปรแกรมส่วนที่2.

ส่วนควบคุมจอภาพ. 3-12

ส่วนขัดการวาดรูปร่างข้อมูลสองมิติ. 13-19.

ส่วนควบคุมดิสก์. 19-26

ใช้ในการพิมพ์อนุกรม. 27.

ส่วนการทดสอบคุณลักษณะ. 32-35

ส่วนขั้นระบบขัดการ (OS. shell) 27-32.

```
                        title project
                        page 45,80
                memory  macro type
                local not_save_upper,not_save_lower,exit_memory
                        push bx
                        push ax
                        mov bl,type
                        cmp bl,1
                        jne not_save_upper
                        dec col
                        mov bx,col
                        mov line1[bx],al
                        inc col
                        jmp exit_memory
                not_save_upper:
                        cmp bl,3
                        jne not_save_lower
                        dec col
                        mov bx,col
                        mov line3[bx],al
                        inc col
                        jmp exit_memory
                not_save_lower:
                        mov bx,col
                        mov line2[bx],al
                        inc pointer
                        inc col
                exit_memory:
                        pop ax
                        pop bx
                        endm

= 00A0                  MaxLine Equ 160 ; Max char on a lines
0000                    code segment
                        assume cs:code ,ds:code
0000                            org 0h
0000                    seg_begin:
0100                            org 100h
0100  E9 136D R         start:  jmp skiphead
                        ; ==============================================
                        ====
```

```
                        ;  data area
0103  ????              First  dw ?
0105  ????              Last dw ?
0107  0000              pointer dw 0  ;use for save end of character
0109  01                mode_ins  db 1  ;1 is in ins
010A  ??                y1 db ?
010B  ??                y2 db ?
010C  ??                     direction db ?         ;use for direction
                        of scroll
010D  01                     mode    db   1        ;1 is in thai mode
                        ,0 is english mode
010E  ??                     noline  db   ?        ;use for scroll up
                        and down macro
010F  01                     linestatus  db   1    ;use for save positi
                        on of thai character
0110  01                     before_char db   1    ;use for save status
                         before character is upper
0111  0000                   col         dw  0     ;use for pointer of m
                        emory buffer
0113  ??                     type        db  ?     ;use in scroll and lo
                        ok_table macro
0114  ????                  cursor_position dw ?
0116  0000                  line            dw 0
0118  ??                     character db ?
0119  ??                     choice db ?
011A  00                     counter_line db 0
011B  0078[                  line1 db 120 dup(' ')
          20
                        ]

0193  0078[                  line2 db 120 dup(' ')
          20
                        ]

020B  0078[                  line3 db 120 dup(' ')
          20
                        ]

0283  EE F3 F7 FB EA E6      table1  db 238,243,247,251,234,230
0289  EF F4 F8 FC EB E7      table2  db 239,244,248,252,235,231
028F  F0 F5 F9 FD EC E8      table3  db 240,245,249,253,236,232
```

```
0295  F1 F6 FA FE ED E9          table4  db 241,246,250,254,237,233
029B  E6 E7 E8 E9                table5  db 230,231,232,233
029F  EA EB EC ED                table6  db 234,235,236,237
02A3  EE EF F0 F1 F2             table7  db 238,239,240,241,242
02A8  F3 F4 F5 F6                table8  db 243,244,245,246
02AC  F7 F8 F9 FA                table9  db 247,248,249,250
02B0  FB FC FD FE                table10 db 251,252,253,254
02B4  0020[                      ascii_table  db 32 dup(0)
                                        ;0-31
          00
                         ]

02D4  20 00 2E 32 33 34 EB                    db 20h ,0  ,2eh ,032h,033h,3
                                4h ,0ebh,0a5h ;32-39
      A5
02DC  36 37 35 39 BF A2 D3                    db 036h,037h,035h,39h ,0bfh,0
                                a2h,0d3h,0bbh ;40-47
      BB
02E4  A6 00 2F 2D BE B4 D7                    db 0a6h,  0h,02fh,02dh,0beh,0
                                b4h,0d7h,0dbh ;48-55
      D8
02EC  A3 B3 A8 C4 B0 A8 C9                    db 0a3h,0b3h,0a8h,0c4h,0b0h,0
                                a8h,0c9h,0bbh ;56-63
      BB
02F4  31 C2 2E A7 AD AC D2                    db 31h ,194 ,2eh ,167 ,173 ,1
                                72 ,210 ,0aah ;64-71
      AA
02FC  DF B1 E3 C6 C5 3F E4                    db 223 ,177 ,227 ,198 ,197 ,6
                                3  ,228 , 214 ;72-79
      D6
0304  AB 30 AF A4 B6 E2 CB                    db 171 ,030h,175 ,164 ,182 ,2
                                26 ,203 ,  34 ;80-87
      22
030C  29 DE 2B B8 5C C3 D8                    db 41  ,222 ,40  ,184 ,92  ,1
                                95 ,216 ,038h ;88-95
      38
0314  A5 BD D9 D1 A1 CF B2                    db 165 ,189 ,217 ,209 ,161 ,2
                                07 ,178 , 208 ;96-103
      D0
031C  E1 C1 E0 CE C7 B5 DC                    db 225 ,193 ,224 ,206 ,199 ,1
                                81 ,220 , 183 ;104-110
```

```
        B7
 0324   C0 D5 BC C8 CC DA CA
                                db 192,213 ,188 ,200 ,204 ,2
        D4                      18 ,202 , 212 ;112-119
 032C   B9 DD BA AE 7C 2C 7E
                                db 185 ,221 ,186 ,174 ,124 ,4
        80                      4  ,126 , 128 ;120-127

                                ; procedue dx use it to modify memory allocate b
                                efore
                                ; allocate other memory
                                ; and allocate momory for first line

 0334                           init proc near
 0334  50                       push ax
 0335  53                       push bx
 0336  57                       push di
 0337  06                       push es
 0338  0E                       push cs
 0339  07                       pop  es
 033A  B4 4A                    mov ah,4ah
 033C  BB 0144                  mov bx,(offset seg_end - offset seg_begin +15 )
                                shr 4
 033F  CD 21                    int 21h
 0341  E8 0363 R                call malloc
 0344  8C C3                    mov bx,es
 0346  89 1E 0103 R             mov First,bx
 034A  89 1E 0105 R             mov last,bx
 034E  33 DB                    xor bx,bx
 0350  8B FB                    mov di,bx
 0352  26: 89 1D                mov es:[di],bx
 0355  26: 89 5D 02             mov es:[di+2],bx
 0359  26: C6 45 04 00          mov byte ptr es:[di+4],0
 035E  07                       pop es
 035F  5B                       pop bx
 0360  5F                       pop di
 0361  58                       pop ax
 0362  C3                       ret
                                init endp
                                ; malloc return new seg in es
 0363                           ; error carry flag set
                                malloc proc near
```

```
0363  50                    push ax
0364  53                    push bx
0365  B4 48                mov ah,48h
0367  BB 000B              mov bx,(MaxLine+5+15)shr 4
036A  CD 21                int 21h
036C  50                    push ax
036D  07                    pop  es
036E  5B                    pop  bx
036F  58                    pop  ax
0370  C3                    ret
                            malloc endp
                            ; input pinter in es
0371                        release proc near
0371  50                    push ax
0372  06                    push es
0373  B4 49                mov ah,49h
0375  CD 21                int 21h
0377  07                    pop  es
0378  58                    pop  ax
0379  C3                    ret
                            release endp
037A                        ErrProc proc near
037A  C3                          ret
                            ErrProc endp
                            ; get old line length
                            ; input bx pointer to the line
                            ; output in ch
037B                        GetLeng proc near
037B  57                          push di
037C  06                          push es
037D  8E C3                       mov es,bx
037F  33 FF                       xor di,di
0381  26: 8A 6D 04                mov ch,es:[di+4]
0385  07                          pop  es
0386  5F                          pop  di
0387  C3                          ret
                            Getleng endp
                            ; Update line length
                            ; input bx pointer to the line ,ch update line l
                            ength
0388                        CngLeng proc near
```

```
038B  57                              push di
0389  06                              push es
038A  8E C3                           mov es,bx
038C  33 FF                           xor di,di
038E  26: 88 6D 04                    mov es:[di+4],ch
0392  07                              pop  es
0393  5F                              pop  di
0394  C3                              ret
                              CngLeng endp
                              ; use to allocate memory for a line and Insert i
                              t to the link lists
                              ; input pointer of line in bx
                              ; output bx will point to new line
0395                          InsLine proc near
0395  50                              push ax
0396  56                              push si
0397  57                              push di
0398  06                              push es
0399  1E                              push ds
039A  E8 0363 R                       call malloc
039D  73 08                           jnc Ins_1
039F  B0 01                           mov al,1
03A1  E8 037A R                       call ErrProc
03A4  EB 30 90                        jmp Ins_Ext
03A7                          Ins_1:
03A7  8E DB                           mov ds,bx ; ds point to update line
03A9  33 FF                           xor di,di ; es point to New Line
03AB  33 F6                           xor si,si ; clear pointer
03AD  8B 44 02                        mov ax,[si+2]
03B0  26: 89 45 02                    mov es:[di+2],ax
03B4  26: C6 45 04 00                 mov byte ptr es:[di+4],0
03B9  8C 44 02                        mov [si+2],es
03BC  26: 89 1D                       mov es:[di],bx
03BF  2E: 3B 1E 0105 R                cmp bx,cs:last
03C4  75 08                           jnz Ins_2
03C6  2E: 8C 06 0105 R                mov cs:last,es
03CB  EB 07 90                        jmp Ins_3
03CE                          Ins_2:
03CE  26: 8E 5D 02                    mov ds,es:[di+2]
03D2  8C 04                           mov [si],es
03D4                          Ins_3:
```

```
03D4  8C C3                     mov bx,es
03D6                        Ins_Ext:
03D6  1F                        pop ds
03D7  07                        pop es
03D8  5F                        pop di
03D9  5E                        pop si
03DA  58                        pop ax
03DB  C3                        ret
                            InsLine endp
                            ;input line to be erase in bx
                            ;output bx point to Nxt line
                            ;       ax point to Pre line
03DC                        DelLine proc near
03DC  56                        push si
03DD  57                        push di
03DE  1E                        push ds
03DF  06                        push es
03E0  8E C3                     mov es,bx
03E2  33 FF                     xor di,di
03E4  33 F6                     xor si,si
03E6  26: 8B 05                 mov ax,es:[di]
03E9  26: 8B 5D 02              mov bx,es:[di+2]
03ED  E8 0371 R                 call release
03F0  8E D8                     mov ds,ax
03F2  89 5C 02                  mov [si+2],bx
03F5  8E C3                     mov es,bx
03F7  26: 89 05                 mov es:[di],ax
03FA  07                        pop es
03FB  1F                        pop ds
03FC  5F                        pop di
03FD  5E                        pop si
03FE  C3                        ret
                            DelLine endp
03FF                        InsBefore proc near
03FF  50                        push ax
0400  56                        push si
0401  57                        push di
0402  1E                        push ds
0403  06                        push es
0404  E8 0363 R                 call malloc
0407  73 08                     jnc InsB_1
```

```
0409  B0 00                          mov al,0
040B  E8 037A R                      call ErrProc
040E  EB 2E 90                       Jmp InsBExit
0411  8E D8                    InsB_1: mov ds,bx
0413  33 F6                          xor si,si
0415  33 FF                          xor di,di
0417  8B 04                          mov ax,[si] ;
0419  26: 89 05                      mov es:[di],ax;New line point to Pre line
041C  26: 8C 5D 02                   mov es:[di+2],ds;New line point to Old line
0420  26: C6 45 04 00          mov byte ptr es:[di+4],0
0425  8C 04                          mov [si],es   ; old line point to New line
0427  2E: 3B 1E 0103 R               cmp bx,cs:first ; Is it first line
042C  75 08                          jnz Insb_2
042E  2E: 8C 06 0103 R               mov cs:first,es
0433  EB 07 90                       jmp Insb_3
0436  26: 8E 1D                Insb_2: mov ds,es:[di]
0439  8C 44 02                       mov [si+2],es;pre line point to new line
043C  8C C3                     Insb_3: mov bx,es
043E                            InsbExit:
043E  07                              pop es
043F  1F                              pop ds
0440  5F                              pop di
0441  5E                              pop si
0442  58                              pop ax
0443  C3                              ret
                                InsBefore endp
                                ; point to Nxt line procedure
                                ; input line pointer in bx,dx = row num
                                ; output  in bx , ZERO FLAG SET WHEN IT IS LAST
                                LINE
                                ;               dx = row num
0444                            NxtLine proc near
0444  06                             push es
0445  57                             push di
0446  8E C3                          mov es,bx
0448  33 FF                          xor di,di
044A  26: 8B 5D 02                   mov bx,es:[di+2]
044E  0B DB                          or bx,bx
0450  75 04                          jnz NxtLine_1
0452  9C                       pushf
0453  8C C3                          mov bx,es
```

```
0455  9D                        popf
0456                            NxtLine_1:
0456  5F                                pop di
0457  07                                pop  es
0458  C3                                ret
                                NxtLine endp
0459                            PreLine proc near
0459  06                                push es
045A  57                                push di
045B  8E C3                             mov es,bx
045D  33 FF                             xor di,di
045F  26: 8B 1D                         mov bx,es:[di]
0462  0B DB                             or bx,bx
0464  75 04                             jnz PreLine_1
0466  9C                        pushf
0467  8C C3                             mov bx,es
0469  9D                        popf
046A                            PreLine_1:
046A  5F                                pop di
046B  07                                pop  es
046C  C3                                ret
                                PreLine endp
                                ; input bx pointer
                                ; input cl order or char
                                ; output es:[di] point to it
                                ; not destroy any register except es and di
                                ; ZERO FLAG SET IF LINE LENG TO LONG
046D                            char_line proc near
046D  51                                push cx
046E  80 F9 A0                          cmp cl,MaxLine
0471  74 09                             jz char_exit
0473  8E C3                             mov es,bx
0475  BF 0005                           mov di,5
0478  32 ED                             xor ch,ch
047A  03 F9                             add di,cx
047C                            char_exit:
047C  59                                pop cx
047D  C3                                ret
                                char_line endp
                                ;Print a decimal number
                                ; input in dx
```

```
047E                    DecPrint proc near
047E  50                        push ax
047F  51                        push cx
0480  52                        push dx
0481  56                        push si
0482  8B C2                     mov ax,dx
0484  BE 000A                   mov si,10
0487  33 C9                     xor cx,cx
0489                    decprint1:
0489  33 D2                     xor dx,dx
048B  F7 F6                     div si
048D  52                        push dx
048E  41                        inc cx
048F  0B C0                     or ax,ax
0491  75 F6                     jne decprint1
0493                    decprint2:
0493  5A                        pop dx
0494  80 C2 30                  add dl,30h
0497  B4 02                     mov ah,2
0499  CD 21                     int 21h
049B  E2 F6                     loop decprint2
049D  5E                        pop si
049E  5A                        pop dx
049F  59                        pop cx
04A0  58                        pop ax
04A1  C3                        ret
04A2                    DecPrint endp
04A2                    scroll proc near
04A2  50                        push ax
04A3  53                        push bx
04A4  51                        push cx
04A5  52                        push dx
04A6  8A 26 010C R              mov ah,direction
04AA  A0 010E R                 mov al,noline
04AD  B7 07                     mov bh,7
04AF  8A 2E 010A R              mov ch,y1
04B3  B1 00                     mov cl,0
04B5  8A 36 010B R              mov dh,y2
04B9  B2 4F                     mov dl,79
04BB  CD 10                     int 10h
04BD  5A                        pop dx
```

```
04BE  59                        pop cx
04BF  5B                        pop bx
04C0  58                        pop ax
04C1  C3                        ret
                        scroll endp
04C2                    getxy proc near
04C2  50                        push ax
04C3  53                        push bx
04C4  51                        push cx
04C5  B4 03                     mov ah,3
04C7  B7 00                     mov bh,0
04C9  CD 10                     int 10h
04CB  59                        pop cx
04CC  5B                        pop bx
04CD  58                        pop ax
04CE  C3                        ret
                        getxy endp
04CF                    gotoxy proc near
04CF  50                        push ax
04D0  53                        push bx
04D1  B7 00                     mov bh,0
04D3  B4 02                     mov ah,2
04D5  CD 10                     int 10h
04D7  5B                        pop bx
04D8  58                        pop ax
04D9  C3                        ret
                        gotoxy endp
                        ;-------------------------
04DA                    printline proc near     ;set position on screen
                         at dx and pointer at si
04DA  50                          push ax
04DB  53                          push bx
04DC  51                          push cx
04DD  52                          push dx
04DE  56                          push si
04DF  E8 04CF R                   call gotoxy
04E2                    again_print:
04E2  8A 84 0193 R                mov al,line2[si]   ;middle character
04E6  E8 04CF R                   call gotoxy
04E9  E8 0FD7 R                   call p_char
04EC  8A 84 011B R                mov al,line1[si]
```

```
04F0  3C 20                     cmp al,20h
04F2  74 0A                     je is_nohaveupper
04F4  FE CE                     dec dh
04F6  E8 04CF R                 call gotoxy
04F9  E8 0FD7 R                 call p_char
04FC  FE C6                     inc dh
04FE                    is_nohaveupper:
04FE  8A 84 020B R              mov al,line3[si]
0502  3C 20                     cmp al,20h
0504  74 0A                     je is_nolower
0506  FE C6                     inc dh
0508  E8 04CF R                 call gotoxy
050B  E8 0FD7 R                 call p_char
050E  FE CE                     dec dh
0510                    is_nolower:
0510  FE C2                     inc dl
0512  46                       inc si
0513  E2 CD                     loop again_print
0515  5E                       pop si
0516  5A                       pop dx
0517  59                       pop cx
0518  5B                       pop bx
0519  58                       pop ax
051A  C3                       ret
                        printline endp
                        ;-------------------------
051B                    clearline proc near     ;set position on screen
                         at dx and pointer at si
051B  50                       push ax
051C  51                       push cx
051D  52                       push dx
051E  56                       push si
                        ;       mov dl,0
                        ;        mov cx,79   ;si is pointer of buffer

051F  E8 04CF R                 call gotoxy
0522                    again_clear:
0522  8A 84 011B R              mov al,line1[si]    ;upper character
0526  3C 20                     cmp al,20h
0528  74 0C                     je is_no_upper
052A  FE CE                     dec dh
```

```
052C  E8 04CF R                 call gotoxy
052F  B0 20                     mov al,20h
0531  E8 0FD7 R                 call p_char
0534  FE C6                     inc dh
0536                        is_no_upper:
0536  8A 84 020B R              mov al,line3[si]     ;lower character
053A  3C 20                     cmp al,20h
053C  74 0C                     je is_no_lower
053E  FE C6                     inc dh
0540  E8 04CF R                 call gotoxy
0543  B0 20                     mov al,20h
0545  E8 0FD7 R                 call p_char
0548  FE CE                     dec dh
054A                        is_no_lower:
054A  FE C2                     inc dl
054C  46                        inc si
054D  E2 D3                     loop again_clear
054F  5E                        pop si
0550  5A                        pop dx
0551  59                        pop cx
0552  58                        pop ax
0553  C3                        ret
                            clearline endp
                            ;--------------------------------
0554                        left proc near
0554  83 3E 0111 R 00           cmp col,0    ;0-119 is 120
0559  75 01                     jne is_con11
055B  C3                        ret
055C                        is_con11:
055C  E8 04C2 R                 call getxy
055F  80 FA 00                  cmp dl,0
0562  75 49                     jne not_write_line
0564  80 3E 010F R 02           cmp linestatus,2        ;2 is upper ,
                            0 is lower ,1 is middle
0569  75 0B                     jne is_1_notupper
056B  FE C6                     inc dh
056D  E8 04CF R                 call gotoxy
0570  EB 0D 90                  jmp 1_con
0573                        is_1_notupper:
0573  80 3E 010F R 00           cmp linestatus,0
0578  75 05                     jne 1_con
```

```
057A  FE CE                             dec dh
057C  E8 04CF R                         call gotoxy
057F                            l_con:
057F  C6 06 010F R 01                   mov linestatus,1
0584  FF 0E 0111 R                      dec col
0588  8B 36 0111 R                      mov si,col
058C  E8 04C2 R                         call getxy
058F  89 16 0114 R                      mov cursor_position,dx
0593  B2 00                             mov dl,0
0595  46                               inc si
0596  B9 004F                          mov cx,79
0599  E8 051B R                         call clearline
059C  4E                               dec si
059D  B2 00                             mov dl,0
059F  B9 004F                          mov cx,79
05A2  E8 04DA R                         call printline    ;si is pointer in b
                                uffer dx is position on screen
05A5  8B 16 0114 R                      mov dx,cursor_position
05A9  E8 04CF R                         call gotoxy
05AC  C3                                ret
05AD                            not_write_line:
05AD  80 3E 010F R 01                   cmp linestatus,1    ;check is middle
05B2  75 03                             jne lnot_middle
05B4  EB 19 90                          jmp lis_middle
05B7                            lnot_middle:
05B7  80 3E 010F R 02                   cmp linestatus,2    ;check is upper ch
                                aracter
05BC  75 0E                             jne lnot_up         ;2 is upper ,0 is
                                lower ,1 is middle
05BE  E8 04C2 R                         call getxy
05C1  FE C6                             inc dh
05C3  E8 04CF R                         call gotoxy
05C6  C6 06 010F R 01                   mov linestatus,1    ;now is in middle
                                line
05CB  C3                                ret
05CC                            lnot_up:
05CC  EB 1B 90                          jmp lcheck_low
05CF                            lis_middle:
05CF  E8 04C2 R                         call getxy
05D2  FE C6                             inc dh
05D4  8B 1E 0111 R                      mov bx,col
```

```
05D8  8A 87 020B R              mov al,line3[bx]
05DC  3C 20                     cmp al,20h
05DE  74 09                     je  lcheck_low
05E0  E8 04CF R                 call gotoxy
05E3  C6 06 010F R 00           mov linestatus,0       ;0 is lower
05E8  C3                        ret
05E9                    lcheck_low:
05E9  FE CE                     dec dh
05EB  FF 0E 0111 R              dec col
05EF  8B 1E 0111 R              mov bx,col
05F3  8A 87 011B R              mov al,line1[bx]
05F7  3C 20                     cmp al,20h
05F9  74 0D                     je puzzle
05FB  FE CE                     dec dh
05FD  FE CA                     dec dl
05FF  E8 04CF R                 call gotoxy
0602  C6 06 010F R 02           mov linestatus,2    ;is upper mode
0607  C3                        ret
0608                    puzzle:
0608  FE CA                     dec dl
060A  E8 04CF R                 call gotoxy
060D  C6 06 010F R 01           mov linestatus,1    ; is middle mode
0612  C3                        ret
                        left endp
                        ;-------------------------
0613                    right proc near
0613  83 3E 0111 R 77           cmp col,119    ;0-119 is 120
0618  75 01                     jne is_con
061A  C3                        ret
061B                    is_con:
061B  A1 0111 R                 mov ax,col
061E  3B 06 0107 R              cmp ax,pointer
0622  72 01                     jb is_conr
0624  C3                        ret
0625                    is_conr:
0625  E8 04C2 R                 call getxy
0628  80 FA 4F                  cmp dl,79
062B  75 4E                     jne r_not_exit
062D  E8 04C2 R                 call getxy
0630  89 16 0114 R              mov cursor_position,dx
0634  80 3E 010F R 02           cmp linestatus,2        ;2 is upper ,
```

```
                        0 is lower ,1 is middle
0639  75 0B                      jne is_r_notupper
063B  FE C6                      inc dh
063D  E8 04CF R                  call gotoxy
0640  EB 0D 90                   jmp r_con
0643                    is_r_notupper:
0643  80 3E 010F R 00              cmp linestatus,0
0648  75 05                      jne r_con
064A  FE CE                      dec dh
064C  E8 04CF R                  call gotoxy
064F                    r_con:
064F  C6 06 010F R 01            mov linestatus,1
0654  8B 1E 0111 R               mov bx,col
0658  83 EB 4E                   sub bx,78
065B  8B F3                      mov si,bx
065D  4E                         dec si
065E  B9 004F                    mov cx,79
0661  B2 00                      mov dl,0
0663  E8 051B R                  call clearline
0666  46                         inc si
0667  B2 00                      mov dl,0
0669  B9 004F                    mov cx,79
066C  E8 04DA R                  call printline     ;si is pointer in b
                        uffer dx is position on screen
066F  8B 16 0114 R               mov dx,cursor_position
0673  E8 04CF R                  call gotoxy
0676  FF 06 0111 R               inc col
067A  C3                         ret
067B                    r_not_exit:
067B  80 3E 010F R 01            cmp linestatus,1   ;check is middle
0680  75 03                      jne rnot_middle
0682  EB 19 90                   jmp ris_middle
0685                    rnot_middle:
0685  80 3E 010F R 02            cmp linestatus,2   ;check is upper ch
                        aracter
068A  75 0E                      jne rnot_up        ;2 is upper ,0 is
                        lower ,1 is middle
068C  E8 04C2 R                  call getxy
068F  FE C6                      inc dh
0691  E8 04CF R                  call gotoxy
0694  C6 06 010F R 01            mov linestatus,1   ;now is in middle
```

```
                        line
0699 C3                         ret
069A                    rnot_up:
069A EB 1B 90                   jmp rcheck_low
069D                    ris_middle:
069D E8 04C2 R                  call getxy
06A0 FE C6                      inc dh
06A2 8B 1E 0111 R               mov bx,col
06A6 8A 87 020B R               mov al,line3[bx]
06AA 3C 20                      cmp al,20h
06AC 74 09                      je  rcheck_low
06AE E8 04CF R                  call gotoxy
06B1 C6 06 010F R 00            mov linestatus,0        ;0 is lower
06B6 C3                         ret
06B7                    rcheck_low:
06B7 FF 06 0111 R               inc col
06BB FE CE                      dec dh
06BD 8B 1E 0111 R               mov bx,col
06C1 8A 87 011B R               mov al,line1[bx]
06C5 3C 20                      cmp al,20h
06C7 74 0D                      je rpuzzle
06C9 FE C2                      inc dl
06CB FE CE                      dec dh
06CD E8 04CF R                  call gotoxy
06D0 C6 06 010F R 02            mov linestatus,2    ;is upper mode
06D5 C3                         ret
06D6                    rpuzzle:
06D6 FE C2                      inc dl
06D8 E8 04CF R                  call gotoxy
06DB C6 06 010F R 01            mov linestatus,1    ; is middle mode
06E0 C3                         ret
                        right endp
                        ;--------------------------------
06E1                    threetoone proc near
06E1 BE 0000                    mov si,0
06E4 C6 06 011A R 00            mov counter_line,0
06E9                    begindata:
06E9 8A 84 0193 R               mov al,line2[si]
06ED 3C 0D                      cmp al,13
06EF 75 08                      jne is_not_cr
06F1 26: 88 05                  mov es:[di],al
```

```
06F4  8A 2E 011A R                    mov ch,counter_line
06F8  E8 03B8 R                       call cngleng
06FB  C3                              ret
06FC                            is_not_cr:
06FC  26: 88 05                        mov es:[di],al     ;save middle charac
                                ter
06FF  47                               inc di
0700  FE 06 011A R                     inc counter_line
0704  8A 84 011B R                     mov al,line1[si]   ;save upper charact
                                er
0708  3C 20                            cmp al,20h
070A  74 08                            je is_no_upperdata
070C  26: 88 05                        mov es:[di],al
070F  47                               inc di
0710  FE 06 011A R                     inc counter_line
0714                            is_no_upperdata:
0714  8A 84 020B R                     mov al,line3[si]
0718  3C 20                            cmp al,20h
071A  74 08                            je is_not_lowerdata
071C  26: 88 05                        mov es:[di],al
071F  47                               inc di
0720  FE 06 011A R                     inc counter_line
0724                            is_not_lowerdata:
0724  46                               inc si
0725  EB C2                           jmp begindata
                                threetoone endp
                                ;---------------------------
0727                            clearbuffer proc near
0727  50                               push ax
0728  51                               push cx
0729  57                               push di
072A  06                               push es
072B  0E                               push cs
072C  07                               pop es
072D  B0 20                            mov al,20h
072F  B9 0168                          mov cx,120*3
0732  BF 011B R                        mov di ,offset line1
0735  F3/ AA                           rep stosb
0737  07                               pop es
0738  5F                               pop di
0739  59                               pop cx
```

```
073A  58                        pop ax
073B  C3                          ret
                        clearbuffer endp
                        ;-------------------------
073C                    onetothree proc near
073C  50                        push ax
073D  51                        push cx
073E  57                        push di
073F  06                        push es
0740  E8 0727 R                 call clearbuffer
0743  E8 037B R                 call getleng
0746  8A CD                     mov cl,ch
0748  B5 00                     mov ch,0
074A  BE 0000                   mov si,0
074D  C7 06 0107 R 0000         mov pointer,0
0753                    put_again:
0753  26: 8A 05                 mov al,es:[di]
0756  3C 0D                     cmp al,13
0758  75 05                     jne is_notcr
075A  07                        pop es
075B  5F                        pop di
075C  59                        pop cx
075D  58                        pop ax
075E  C3                        ret
075F                    is_notcr:
075F  3C D7                     cmp al,215
0761  72 16                     jb is_middlelower
0763  3C D9                     cmp al,217
0765  73 09                     jae is_upper
0767  4E                        dec si
0768  88 84 020B R               mov line3[si],al        ;put lower chara
                        cter
076C  46                        inc si
076D  47                        inc di
076E  EB E3                     jmp put_again
0770                    is_upper:
0770  4E                        dec si
0771  88 84 011B R               mov line1[si],al        ;put upper chara
                        cter
0775  46                        inc si
0776  47                        inc di
```

```
0777  EB DA                        jmp put_again
0779                          is_middlelower:
0779  88 84 0193 R                 mov line2[si],al
077D  46                            inc si
077E  47                            inc di
077F  FF 06 0107 R                  inc pointer
0783  EB CE                         jmp put_again
                              onetothree endp
                              ;----------------------
0785                          return proc near
0785  80 3E 010F R 01               cmp linestatus,1
078A  75 03                         jne renot_middle
078C  EB 25 90                      jmp re_middle
078F                          renot_middle:
078F  80 3E 010F R 02               cmp linestatus,2   ;is upper
0794  75 10                         jne renot_up
0796  E8 04C2 R                     call getxy
0799  FE C6                         inc dh
079B  E8 04CF R                     call gotoxy
079E  C6 06 010F R 01               mov linestatus,1
07A3  EB 0E 90                      jmp re_middle
07A6                          renot_up:
07A6  E8 04C2 R                     call getxy
07A9  FE CE                         dec dh
07AB  E8 04CF R                     call gotoxy
07AE  C6 06 010F R 01               mov linestatus,1
07B3                          re_middle:
07B3  80 3E 0109 R 01               cmp mode_ins,1   ;check is insert mode
07B8  75 04                         jne isre_overwrite
07BA  E8 0841 R                     call returnins   ;routine to return at in
                              sert mode
07BD  C3                            ret
                              ;----------use for retrun at overwrite mode
07BE                          isre_overwrite:
                              ;--------save data from three line to link list
07BE  8B 1E 0116 R                  mov bx,line
07C2  B1 00                         mov cl,0
07C4  E8 046D R                     call char_line
07C7  8B 36 0107 R                  mov si,pointer
07CB  B0 0D                         mov al,0dh
07CD  88 84 0193 R                  mov line2[si],al   ;save cr into three li
```

```
                        ne buffer
07D1  E8 06E1 R                 call threetoone
                        ;---------save data from three line to link list

07D4  8B 1E 0116 R              mov bx,line
07D8  E8 0444 R                 call nxtline      ;line is 028c
07DB  75 01                     jnz is_have_last
07DD  C3                        ret
07DE                    is_have_last:
                        ;-----------read data form next line
07DE  89 1E 0116 R              mov line,bx
07E2  B1 00                     mov cl,0
07E4  E8 0460 R                 call char_line
07E7  E8 0727 R                 call clearbuffer
07EA  E8 073C R                 call onetothree
                        ;-----------read data form next line
07ED  C7 06 0111 R 0000         mov col,0
07F3  E8 04C2 R                 call getxy
07F6  FE C6                     inc dh
07F8  FE C6                     inc dh
07FA  FE C6                     inc dh
07FC  80 FE 17                  cmp dh,23
07FF  77 06                     ja rmor23
0801  B2 00                     mov dl,0
0803  E8 04CF R                 call gotoxy
0806  C3                        ret
0807                    rmor23:
                        ;--------scroll screen and set cursor
0807  C6 06 010C R 06           mov direction,6
080C  C6 06 010E R 03           mov noline,3
0811  C6 06 010A R 00           mov y1,0
0816  C6 06 010B R 18           mov y2,24
081B  E8 04A2 R                 call scroll  ;scroll_up 3 line
                        ;--------scroll screen and set cursor
081E  B6 17                     mov dh,23
0820  B2 00                     mov dl,0
0822  52                       push dx
0823  52                       push dx
0824  8B 0E 0107 R             mov cx,pointer
0828  83 F9 00                  cmp cx,0    ;use for next line have not c
haracter
```

```
0B2B  74 0F                     je is_cx_zero
0B2D  B3 F9 50                  cmp cx,80
0830  7C 03                     jl is_less80
0832  B9 0050                   mov cx,80
0835                        is_less80:
0835  BE 0000                   mov si,0
0838  5A                       pop dx
0839  E8 04DA R                call printline
083C                        is_cx_zero:
083C  5A                       pop dx
083D  E8 04CF R                call gotoxy
0840  C3                       ret
                            return endp
                            ;--------------------------
0841                        returnins proc near
0841  A1 0107 R                mov ax,pointer
0844  39 06 0111 R             cmp col,ax
0848  74 03                    je is_returnok
084A  E9 0905 R                jmp r_t
084D                        is_returnok:
                            ;-------------------scroll screen
084D  E8 04C2 R                call getxy
0850  80 FE 17                 cmp dh,23
0853  75 1A                    jne isn_y_23
0855  C6 06 010B R 18          mov y2,24
085A  C6 06 010C R 06          mov direction,6
085F  C6 06 010E R 03          mov noline,3
0864  C6 06 010A R 00          mov y1,0
0869  E8 04A2 R                call scroll
086C  EB 23 90                 jmp cr_con
086F                        isn_y_23:
086F  B0 03                     mov al,3
0871  80 FE 14                 cmp dh,20
0874  75 02                    jne isn_y20
0876  B0 02                    mov al,2
0878                        isn_y20:
0878  A2 010E R                mov noline,al
087B  80 C6 02                 add dh,2
087E  C6 06 010C R 07          mov direction,7
0883  88 36 010A R             mov y1,dh
0887  C6 06 010B R 18          mov y2,24
```

```
088C  E8 04A2 R                       call scroll   ;scroll_up 3 line
088F  FE C6                           inc dh
                                    ;------------------scrooll screen
                                    ;---------set position cursor and save cr in thr
                                    ee line buffer
0891                                cr_con:
0891  32 D2                           xor dl,dl
0893  E8 04CF R                       call gotoxy
0896  B0 0D                           mov al,0dh
                                      memory 2
0898  53                       1      push bx
0899  50                       1      push ax
089A  B3 02                     1      mov bl,2
089C  80 FB 01                  1      cmp bl,1
089F  75 13                     1      jne ??0000
08A1  FF 0E 0111 R              1      dec col
08A5  8B 1E 0111 R              1      mov bx,col
08A9  88 87 011B R              1      mov line1[bx],al
08AD  FF 06 0111 R              1      inc col
08B1  EB 29 90                  1      jmp ??0002
08B4                            1  ??0000:
08B4  80 FB 03                  1      cmp bl,3
08B7  75 13                     1      jne ??0001
08B9  FF 0E 0111 R              1      dec col
08BD  8B 1E 0111 R              1      mov bx,col
08C1  88 87 020B R              1      mov line3[bx],al
08C5  FF 06 0111 R              1      inc col
08C9  EB 11 90                  1      jmp ??0002
08CC                            1  ??0001:
08CC  8B 1E 0111 R              1      mov bx,col
08D0  88 87 0193 R              1      mov line2[bx],al
08D4  FF 06 0107 R              1      inc pointer
08D8  FF 06 0111 R              1      inc col
08DC                            1  ??0002:
08DC  58                        1      pop ax
08DD  5B                        1      pop bx
                                    ;---------set position cursor and save cr in thr
                                    ee line buffer
                                    ;---------save data form three to link list
08DE  8B 1E 0116 R                    mov bx,line
08E2  B1 00                           mov cl,0
```

```
08E4  E8 046D R                 call char_line   ;calculate pointer at l
                              ink list
08E7  E8 06E1 R                 call threetoone  ;save to link list
                              ;---------save data form three to link list
08EA  8B 1E 0116 R              mov bx,line
08EE  E8 0395 R                 call insline     ;line is 028c
08F1  89 1E 0116 R              mov line,bx
08F5  C7 06 0111 R 0000         mov col,0
08FB  C7 06 0107 R 0000         mov pointer,0
0901  E8 0727 R                 call clearbuffer
0904  C3                        ret
0905                          r_t:
0905  E8 04C2 R                 call getxy
0908  52                       push dx
                              ;-----------use for clear append line at screen
0909  8B 0E 0107 R             mov cx,pointer
090D  2B 0E 0111 R             sub cx,col
0911  B0 20                    mov al,20h
0913                          clear_append:
0913  E8 04CF R                 call gotoxy
0916  E8 0FD7 R                 call p_char
0919  FE C6                    inc dh
091B  E8 04CF R                 call gotoxy
091E  E8 0FD7 R                 call p_char
0921  FE CE                    dec dh
0923  FE CE                    dec dh
0925  E8 04CF R                 call gotoxy
0928  E8 0FD7 R                 call p_char
092B  FE C6                    inc dh
092D  FE C2                    inc dl
092F  E2 E2                    loop clear_append
                              ;----------use for clear append line at screen
0931  5A                       pop dx   ;save cursor position
                              ;-------------------scroll screen
0932  80 FE 17                 cmp dh,23
0935  75 1A                    jne risn_y_23
0937  C6 06 010B R 18          mov y2,24
093C  C6 06 010C R 06          mov direction,6
0941  C6 06 010E R 03          mov noline,3          ;at line less 20
                              scroll 3 all screen
0946  C6 06 010A R 00          mov y1,0
```

```
094B  E8 04A2 R                 call scroll
094E  EB 23 90                  jmp cr_con1
0951                        risn_y_23:
0951  B0 03                       mov al,3
0953  80 FE 14                  cmp dh,20
0956  75 02                     jne risn_y20      ;at line 20 scroll 2 li
                            ne
0958  B0 02                     mov al,2
095A                        risn_y20:
095A  A2 010E R                 mov noline,al
095D  80 C6 02                  add dh,2
0960  C6 06 010C R 07           mov direction,7       ;at
0965  88 36 010A R              mov y1,dh
0969  C6 06 010B R 18           mov y2,24
096E  EB 04A2 R                 call scroll   ;scroll_up 3 line
0971  FE C6                     inc dh
                            ;-------------------scroll screen
0973                        cr_con1:
0973  32 D2                     xor dl,dl
0975  E8 04CF R                 call gotoxy
0978  8B 1E 0116 R              mov bx,line
097C  B1 00                     mov cl,0
097E  E8 046D R                 call char_line
0981  8B 0E 0111 R              mov cx,col
0985  BE 0000                   mov si,0
0988  C6 06 011A R 00           mov counter_line,0
098D  83 F9 00                  cmp cx,0
0990  74 2F                     je is_exit_cr
                            ;save character form three to link list
0992                        save_rot:
0992  8A 84 0193 R              mov al,line2[si]
0996  26: 88 05                 mov es:[di],al
0999  FE 06 011A R              inc counter_line
099D  47                        inc di
099E  8A 84 020B R              mov al,line3[si]
09A2  3C 20                     cmp al,20h
09A4  74 08                     je is_conr1
09A6  26: 88 05                 mov es:[di],al
09A9  FE 06 011A R              inc counter_line
09AD  47                        inc di
09AE                        is_conr1:
```

```
09AE  8A 84 011B R              mov al,line1[si]
09B2  3C 20                     cmp al,20h
09B4  74 08                     je is_conri
09B6  26: 88 05                 mov es:[di],al
09B9  FE 06 011A R              inc counter_line
09BD  47                        inc di
09BE                        is_conri:
09BE  46                        inc si
09BF  E2 D1                     loop save_rot
                            ;save character in three to link list
09C1                        is_exit_cr:
09C1  B0 0D                     mov al,0dh
09C3  26: 88 05                 mov es:[di],al  ;save character return to
                             line
09C6  A1 0107 R                 mov ax,pointer
09C9  8B C8                     mov cx,ax    ;have bug
09CB  51                        push cx      ;use for clear buffer
09CC  2B 06 0111 R              sub ax,col
09D0  8B C8                     mov cx,ax
09D2  8B 36 0111 R              mov si,col
09D6  BF 0000                   mov di,0
09D9  89 0E 0107 R              mov pointer,cx          ;set pointer
09DD                        re_loop:
09DD  8A 84 0193 R              mov al,line2[si]
09E1  88 85 0193 R              mov line2[di],al
09E5  8A 84 020B R              mov al,line3[si]
09E9  88 85 020B R              mov line3[di],al
09ED  8A 84 011B R              mov al,line1[si]
09F1  88 85 011B R              mov line1[di],al
09F5  47                        inc di
09F6  46                        inc si
09F7  E2 E4                     loop re_loop
09F9  59                        pop cx
09FA  B0 20                     mov al,20h
09FC                        clear:                          ;loop clear m
                            emory
09FC  88 85 011B R              mov line1[di],al
0A00  88 85 0193 R              mov line2[di],al
0A04  88 85 020B R              mov line3[di],al
0A08  47                        inc di
0A09  E2 F1                     loop clear
```

```
0A0B  BE 0000                   mov si,0
0A0E  8B 0E 0107 R              mov cx,pointer
0A12  B2 00                     mov dl,0
0A14  E8 04DA R                 call printline
0A17  8B 1E 0116 R              mov bx,line
0A1B  E8 0395 R                 call insline      ;line is 028c
0A1E  89 1E 0116 R              mov line,bx
0A22  C7 06 0111 R 0000         mov col,0       ;set col at zero
0A2B  B2 00                     mov dl,0
0A2A  E8 04CF R                 call gotoxy    ;set cursor position at be
                        gin line
0A2D  C3                        ret
                        returnins endp
0A2E                    up proc near
0A2E  80 3E 010F R 01             cmp linestatus,1
0A33  75 03                       jne unot_middle
0A35  EB 25 90                    jmp uis_middle
0A38                    unot_middle:
0A38  80 3E 010F R 02             cmp linestatus,2
0A3D  75 10                       jne unot_up
0A3F  E8 04C2 R                   call getxy
0A42  FE C6                       inc dh
0A44  E8 04CF R                   call gotoxy
0A47  C6 06 010F R 01             mov linestatus,1
0A4C  EB 0E 90                    jmp uis_middle
0A4F                    unot_up:
0A4F  E8 04C2 R                   call getxy
0A52  FE CE                       dec dh
0A54  E8 04CF R                   call gotoxy
0A57  C6 06 010F R 01             mov linestatus,1
0A5C                    uis_middle:
                        ;------use to save current line in link list
0A5C  8B 36 0107 R                mov si,pointer
0A60  B0 0D                       mov al,0dh
0A62  88 84 0193 R                mov line2[si],al
0A66  8B 1E 0116 R                mov bx,line
0A6A  B1 00                       mov cl,0
0A6C  E8 046D R                   call char_line
0A6F  E8 06E1 R                   call threetoone
                        ;-------------------------
0A72  E8 04C2 R                   call getxy
```

```
0A75  80 FE 03                cmp dh,3
0A78  7E 35                   jle u_problem
0A7A  8B 1E 0116 R            mov bx,line
0A7E  E8 0459 R               call preline
0A81  75 01                   jnz is_not_first
0A83  C3                      ret
0A84                        is_not_first:
0A84  89 1E 0116 R            mov line,bx
0A88  B1 00                   mov cl,0
0A8A  E8 046D R               call char_line
0A8D  E8 073C R               call onetothree
0A90  A1 0107 R               mov ax,pointer
0A93  3B 06 0111 R            cmp ax,col
0A97  73 0C                   jae is_conllll
0A99  A3 0111 R               mov col,ax        ;pointer less than col
0A9C  8A D0                   mov dl,al
0A9E  3D 004F                 cmp ax,79
0AA1  76 02                   jbe is_conllll
0AA3  B2 4F                   mov dl,79
0AA5                        is_conllll:   ;pointer more than col
0AA5  FE CE                   dec dh
0AA7  FE CE                   dec dh
0AA9  FE CE                   dec dh
0AAB  E8 04CF R               call gotoxy
0AAE  C3                      ret
0AAF                        u_problem:
0AAF  E8 04CF R               call gotoxy
0AB2  52                      push dx
0AB3  52                      push dx
0AB4  8B 1E 0116 R            mov bx,line
0AB8  E8 0459 R               call preline
0ABB  75 03                   jnz is_not_first1
0ABD  5A                      pop dx
0ABE  5A                      pop dx
0ABF  C3                      ret
0AC0                        is_not_first1:
0AC0  C6 06 010B R 1B         mov y2,24
0AC5  C6 06 010C R 07         mov direction,7
0ACA  C6 06 010E R 03         mov noline,3          ;at line less 20 scro
                            ll 3 all screen
0ACF  C6 06 010A R 01         mov y1,1
```

```
0AD4  E8 04A2 R                 call scroll
0AD7  89 1E 0116 R              mov line,bx
0ADB  B1 00                     mov cl,0
0ADD  E8 046D R                 call char_line
0AE0  E8 073C R                 call onetothree
0AE3  BE 0000                   mov si,0
0AE6  8B 0E 0107 R              mov cx,pointer
0AEA  5A                        pop dx
0AEB  B2 00                     mov dl,0
0AED  83 F9 00                  cmp cx,0
0AF0  74 1D                     je is_cannot
0AF2  E8 04DA R                 call printline
0AF5  5A                        pop dx
0AF6  A1 0107 R                  mov ax,pointer
0AF9  3B 06 0111 R              cmp ax,col
0AFD  73 0C                     jae is_con2222
0AFF  A3 0111 R                 mov col,ax        ;pointer less than col
0B02  8A D0                     mov dl,al
0B04  3D 004F                   cmp ax,79
0B07  76 02                     jbe is_con2222
0B09  B2 4F                     mov dl,79
0B0B                          is_con2222:    ;pointer more than col
0B0B  E8 04CF R                 call gotoxy
0B0E  C3                        ret
0B0F                          is_cannot:
0B0F  5A                         pop dx
0B10  C3                         ret
                              up endp
                              ;---------------------------
0B11                          down proc near
0B11  80 3E 010F R 01           cmp linestatus,1
0B16  75 03                     jne unot_middle2
0B18  EB 25 90                  jmp uis_middle2
0B1B                          unot_middle2:
0B1B  80 3E 010F R 02           cmp linestatus,2
0B20  75 10                     jne unot_up2
0B22  E8 04C2 R                 call getxy
0B25  FE C6                     inc dh
0B27  E8 04CF R                 call gotoxy
0B2A  C6 06 010F R 01           mov linestatus,1
0B2F  EB 0E 90                  jmp uis_middle2
```

```
0B32                            unot_up2:
0B32  E8 04C2 R                   call getxy
0B35  FE CE                       dec dh
0B37  E8 04CF R                   call gotoxy
0B3A  C6 06 010F R 01             mov linestatus,1
0B3F                            uis_middle2:
                                ;------use to save current line in link list
0B3F  8B 36 0107 R                mov si,pointer
0B43  B0 0D                       mov al,0dh
0B45  88 84 0193 R                mov line2[si],al
0B49  8B 1E 0116 R                mov bx,line
0B4D  B1 00                       mov cl,0
0B4F  E8 046D R                   call char_line
0B52  E8 06E1 R                   call threetoone
                                ;-------------------------
0B55  E8 04C2 R                   call getxy
0B58  80 FE 17                    cmp dh,23
0B5B  73 32                       jae u_problem2
0B5D  8B 1E 0116 R                mov bx,line
0B61  E8 0444 R                   call nxtline
0B64  75 01                       jnz is_not_first2
0B66  C3                          ret
0B67                            is_not_first2:
0B67  89 1E 0116 R                mov line,bx
0B6B  B1 00                       mov cl,0
0B6D  E8 046D R                   call char_line
0B70  E8 073C R                   call onetothree
0B73  80 C6 03                    add dh,3
0B76  A1 0107 R                   mov ax,pointer
0B79  3B 06 0111 R                cmp ax,col
0B7D  73 0C                       jae is_con3333
0B7F  A3 0111 R                   mov col,ax        ;pointer less than col
0B82  8A D0                       mov dl,al
0B84  3D 004F                     cmp ax,79
0B87  76 02                       jbe is_con3333
0B89  B2 4F                       mov dl,79
0B8B                            is_con3333:    ;pointer more than col
0B8B  E8 04CF R                   call gotoxy
0B8E  C3                          ret
                                ;-------use for witre lower line at the bottom s
                                creen
```

```
0B8F                    u_problem2:
0B8F  E8 04CF R           call gotoxy
0B92  52                  push dx
0B93  52                  push dx
0B94  8B 1E 0116 R        mov bx,line
0B98  E8 0444 R           call nxtline
0B9B  75 03               jnz is_first2    ;if last line exit from subr
                        outine
0B9D  5A                  pop dx
0B9E  5A                  pop dx
0B9F  C3                  ret
0BA0                    is_first2:
0BA0  C6 06 010B R 18     mov y2,24
0BA5  C6 06 010C R 06     mov direction,6
0BAA  C6 06 010E R 03     mov noline,3          ;at line less 20 scro
                        ll 3 all screen
0BAF  C6 06 010A R 01     mov y1,1
0BB4  E8 04A2 R           call scroll
0BB7  89 1E 0116 R        mov line,bx
0BBB  B1 00               mov cl,0
0BBD  E8 046D R           call char_line
0BC0  E8 073C R           call onetothree
0BC3  BE 0000             mov si,0
0BC6  8B 0E 0107 R        mov cx,pointer
0BCA  5A                  pop dx
0BCB  B2 00               mov dl,0
0BCD  83 F9 00            cmp cx,0
0BD0  76 1D               jbe is_cannot1
0BD2  E8 04DA R           call printline
0BD5  5A                  pop dx
0BD6  A1 0107 R           mov ax,pointer
0BD9  3B 06 0111 R        cmp ax,col
0BDD  73 0C               jae is_con4444
0BDF  A3 0111 R           mov col,ax       ;pointer less than col
0BE2  8A D0               mov dl,al
0BE4  3D 004F             cmp ax,79
0BE7  76 02               jbe is_con4444
0BE9  B2 4F               mov dl,79
0BEB                    is_con4444:    ;pointer more than col
0BEB  E8 04CF R           call gotoxy
0BEE  C3                  ret
```

```
0BEF                    is_cannot1:
0BEF  5A                    pop dx
0BF0  C3                    ret
                        down endp
0BF1                    rubout proc near
0BF1  80 3E 010F R 01           cmp linestatus,1
0BF6  75 03                     jne rubnot_middle
0BF8  EB 25 90                  jmp rub_middle
0BFB                    rubnot_middle:
0BFB  80 3E 010F R 02           cmp linestatus,2   ;is upper
0C00  75 10                     jne rubnot_up
0C02  E8 04C2 R                 call getxy
0C05  FE C6                     inc dh
0C07  E8 04CF R                 call gotoxy
0C0A  C6 06 010F R 01           mov linestatus,1
0C0F  EB 0E 90                  jmp rub_middle
0C12                    rubnot_up:
0C12  E8 04C2 R                 call getxy
0C15  FE CE                     dec dh
0C17  E8 04CF R                 call gotoxy
0C1A  C6 06 010F R 01           mov linestatus,1
0C1F                    rub_middle:
0C1F  83 3E 0111 R 00           cmp col;0
0C24  75 01                     jne is_not_begin
                        ;        call rubout_begin
0C26  C3                        ret
0C27                    is_not_begin:
0C27  A1 0107 R                 mov ax,pointer
0C2A  8B 1E 0111 R              mov bx,col
0C2E  40                        inc ax
0C2F  2B C3                     sub ax,bx
0C31  8B C8                     mov cx,ax
0C33  E8 04C2 R                 call getxy
0C36  80 FA 00                  cmp dl,0
0C39  75 02                     jne is_not_end
0C3B  FE C2                     inc dl
0C3D                    is_not_end:
0C3D  FE CA                     dec dl
0C3F  52                        push dx
0C40  8B 36 0111 R              mov si,col
0C44  4E                        dec si
```

```
0C45  E8 0518 R                 call clearline
0C48  46                        inc si
0C49  E8 040A R                 call printline
0C4C  5A                        pop dx
0C4D  E8 04CF R                 call gotoxy
0C50  8B 36 0111 R              mov si,col
0C54  8B FE                     mov di,si
0C56  4F                        dec di
0C57                        again_rub:
0C57  8A 84 0193 R              mov al,line2[si]
0C5B  88 85 0193 R              mov line2[di],al
0C5F  8A 84 020B R              mov al,line3[si]
0C63  88 85 020B R              mov line3[di],al
0C67  8A 84 011B R              mov al,line1[si]
0C6B  88 85 011B R              mov line1[di],al
0C6F  46                        inc si
0C70  47                        inc di
0C71  E2 E4                     loop again_rub
0C73  FF 0E 0107 R              dec pointer
0C77  FF 0E 0111 R              dec col
0C7B  C3                        ret
0C7C                        rubout_begin proc near
0C7C  B1 00                     mov cl,0
0C7E  E8 046D R                 call char_line
0C81  E8 06E1 R                 call threetoone  ;save current line into
                            line buffer
0C84  E8 037B R                 call getleng     ;get current line of cha
                            racter in ch register
0C87  8A C5                     mov al,ch
0C89  8B 1E 0116 R              mov bx,line
0C8D  E8 0459 R                 call preline
0C90  89 1E 0116 R              mov line,bx
0C94  E8 037B R                 call getleng   ;get pre line of character

0C97  02 C5                     add al,ch
0C99  3C A0                     cmp al,160
0C9B  76 01                     jbe is_rub_con
0C9D  C3                        ret
0C9E                        is_rub_con:
0C9E  E8 0727 R                 call clearbuffer   ;clear buffer before t
                            o load data form
```

```
0CA1  8B 1E 0116 R              mov bx,line
0CA5  B1 00                     mov cl,0
0CA7  E8 046D R                 call char_line
0CAA                        beginrub:
0CAA  26: 8A 05                 mov al,es:[di]
0CAD  3C 0D                     cmp al,0dh
0CAF  75 03                     jne is_not_out
0CB1  EB 29 90                  jmp exitrubout
0CB4                        is_not_out:
0CB4  3C D9                     cmp al,217
0CB6  72 07                     jb  isnot_upper
0CB8  88 84 011B R              mov line1[si],al
0CBC  47                        inc di
0CBD  EB EB                     jmp beginrub
0CBF                        isnot_upper:
0CBF  3C D7                     cmp al,215
0CC1  72 07                     jb isnot_lower
0CC3  88 84 020B R              mov line3[si],al
0CC7  47                        inc di
0CC8  EB E0                     jmp beginrub
0CCA                        isnot_lower:
0CCA  88 84 0193 R              mov line2[si],al
0CCE  46                        inc si
0CCF  47                        inc di
0CD0  FE C2                     inc dl
0CD2  FF 06 0111 R              inc col
0CD6  FF 06 0107 R              inc pointer
0CDA  EB CE                     jmp beginrub
0CDC                        exitrubout:
0CDC  E8 04CF R                 call gotoxy
0CDF  8B 1E 0116 R              mov bx,line
0CE3  E8 0444 R                 call nxtline
0CE6  89 1E 0116 R              mov line,bx
0CEA  B1 00                     mov cl,0
0CEC  E8 046D R                 call char_line
0CEF  8B 36 0107 R              mov si,pointer
0CF3                        beginrub1:
0CF3  26: 8A 05                 mov al,es:[di]
0CF6  3C 00                     cmp al,0d
0CFB  74 1E                     je exit1
0CFA  3C D9                     cmp al,217
```

```
0CFC  72 07                     jb isn_upper
0CFE  88 84 011B R              mov line1[si],al
0D02  47                        inc di
0D03  EB EE                     jmp beginrub1
0D05                        isn_upper:
0D05  3C D7                      cmp al,215
0D07  72 07                      jb isn_lower
0D09  88 84 020B R               mov line3[si],al
0D0D  47                         inc di
0D0E  EB E3                      jmp beginrub1
0D10                        isn_lower:
0D10  88 84 0193 R               mov line2[si],al
0D14  47                         inc di
0D15  46                         inc si
0D16  EB DB                      jmp beginrub1
0D18                        exit1:
0D18  C3                         ret
                            rubout_begin endp
                            rubout endp
                            ;-------------------------
0D19                        delete proc near
0D19  80 3E 010F R 01            cmp linestatus,1
0D1E  75 03                      jne dsnot_middle
0D20  EB 25 90                   jmp dsis_middle
0D23                        dsnot_middle:
0D23  80 3E 010F R 02            cmp linestatus,2   ;is upper
0D28  75 10                      jne dsnot_up
0D2A  E8 04C2 R                  call getxy
0D2D  FE C6                      inc dh
0D2F  E8 04CF R                  call gotoxy
0D32  C6 06 010F R 01            mov linestatus,1
0D37  EB 0E 90                   jmp dsis_middle
0D3A                        dsnot_up:
0D3A  E8 04C2 R                  call getxy
0D3D  FE CE                      dec dh
0D3F  E8 04CF R                  call gotoxy
0D42  C6 06 010F R 01            mov linestatus,1
0D47                        dsis_middle:
0D47  A1 0107 R                  mov ax,pointer
0D4A  8B 1E 0111 R               mov bx,col
0D4E  3B D8                      cmp bx,ax
```

```
0D50  72 01                     jb is_con_del
0D52  C3                        ret
0D53                        is_con_del:
0D53  2B C3                     sub ax,bx
0D55  8B C8                     mov cx,ax
0D57  E8 04C2 R                 call getxy
0D5A  52                        push dx
0D5B  8B 36 0111 R              mov si,col
0D5F  E8 051B R                 call clearline
0D62  46                        inc si
0D63  E8 04DA R                 call printline
0D66  5A                        pop dx
0D67  E8 04CF R                 call gotoxy
0D6A  8B 36 0111 R              mov si,col
0D6E  8B FE                     mov di,si
0D70  46                        inc si
0D71                        again_delete:
0D71  8A 84 0193 R              mov al,line2[si]
0D75  88 85 0193 R              mov line2[di],al
0D79  8A 84 020B R              mov al,line3[si]
0D7D  88 85 020B R              mov line3[di],al
0D81  8A 84 011B R              mov al,line1[si]
0D85  88 85 011B R              mov line1[di],al
0D89  46                        inc si
0D8A  47                        inc di
0D8B  E2 E4                     loop again_delete
0D8D  FF 0E 0107 R              dec pointer
0D91  C3                        ret
                            delete endp
                            ;---------------------------

0D92                        shift proc near
0D92  E8 04C2 R                 call getxy
0D95  52                        push dx
0D96  FE C2                     inc dl
0D98  E8 04CF R                 call gotoxy
0D9B  A1 0107 R                 mov ax,pointer
0D9E  8B 1E 0111 R              mov bx,col
0DA2  2B C3                     sub ax,bx
0DA4  8B C8                     mov cx,ax
0DA6  51                        push cx
```

```
0DA7                            again_shift:
0DA7  80 FA 4F                        cmp dl,79
0DAA  72 03                           jb is_less79
0DAC  EB 2C 90                        jmp exit_loop
0DAF                            is_less79:
0DAF  8A 87 0193 R                    mov al,line2[bx]
0DB3  E8 0FD7 R                       call p_char
0DB6  8A 87 011B R                    mov al,line1[bx]
0DBA  FE CE                           dec dh
0DBC  E8 04CF R                       call gotoxy
0DBF  E8 0FD7 R                       call p_char
0DC2  8A 87 020B R                    mov al,line3[bx]
0DC6  FE C6                           inc dh
0DC8  FE C6                           inc dh
0DCA  E8 04CF R                       call gotoxy
0DCD  E8 0FD7 R                       call p_char
0DD0  FE C2                           inc dl
0DD2  FE CE                           dec dh
0DD4  E8 04CF R                       call gotoxy
0DD7  43                              inc bx
0DD8  E2 CD                           loop again_shift
0DDA                            exit_loop:
0DDA  59                              pop cx
0DDB  8B 36 0107 R                    mov si,pointer
0DDF  4E                              dec si
0DE0  8B 3E 0107 R                    mov di,pointer
0DE4                            shift1:
0DE4  8A 84 0193 R                    mov al,line2[si]
0DE8  88 85 0193 R                    mov line2[di],al
0DEC  8A 84 020B R                    mov al,line3[si]
0DF0  88 85 020B R                    mov line3[di],al
0DF4  8A 84 011B R                    mov al,line1[si]
0DF8  88 85 011B R                    mov line1[di],al
0DFC  4E                              dec si
0DFD  4F                              dec di
0DFE  E2 E4                           loop shift1
0E00  5A                              pop dx
0E01  E8 04CF R                       call gotoxy
0E04  C3                              ret
                                shift endp
0E05                            print_english proc near
```

```
0E05  50                              push ax
0E06  E8 04C2 R                       call getxy
0E09  80 3E 010F R 02                 cmp linestatus,2
0E0E  75 0D                           jne is_n_e_up
0E10  FE C6                           inc dh
0E12  E8 04CF R                       call gotoxy
0E15  C6 06 010F R 01                 mov linestatus,1
0E1A  EB 12 90                        jmp begin_eng
0E1D                            is_n_e_up:
0E1D  80 3E 010F R 00                 cmp linestatus,0
0E22  75 0A                           jne begin_eng
0E24  FE CE                           dec dh
0E26  E8 04CF R                       call gotoxy
0E29  C6 06 010F R 01                 mov linestatus,1
0E2E                            begin_eng:
0E2E  A1 0107 R                       mov ax,pointer
0E31  8B 1E 0111 R                    mov bx,col
0E35  3B D8                           cmp bx,ax
0E37  74 0A                           je is_overwrite
0E39  80 3E 0109 R 01                 cmp mode_ins,1  ;1 is ins mode
0E3E  75 03                           jne is_overwrite
0E40  E8 0D92 R                       call shift
0E43                            is_overwrite:
0E43  E8 04C2 R                       call getxy
0E46  80 FA 4F                        cmp dl,79
0E49  75 23                           jne is_con1
0E4B  8B 1E 0111 R                    mov bx,col
0E4F  83 EB 4E                        sub bx,78
0E52  8B F3                           mov si,bx
0E54  E8 04C2 R                       call getxy
0E57  B2 00                           mov dl,0
0E59  4E                              dec si
0E5A  B9 004F                         mov cx,79
0E5D  E8 051B R                       call clearline          ;subroutine t
                                o clear upper and lower char
0E60  46                              inc si  ;use si to pointer start
0E61  B9 004F                         mov cx,79
0E64  B2 00                           mov dl,0
0E66  E8 04DA R                       call printline     ;si is pointer in b
                                uffer dx is position on screen
0E69  B2 4E                           mov dl,78    ;set cursor in this posit
```

```
                        ion
0E6B  E8 04CF R                 call gotoxy
0E6E                    is_con1:
0E6E  E8 04C2 R                 call getxy
0E71  89 16 0114 R              mov cursor_position,dx
0E75  58                        pop ax
0E76  E8 0FD7 R                 call p_char          ;is in english mo
                        de
0E79  A2 0118 R                 mov character,al
0E7C  C6 06 0119 R 01           mov choice,1
0E81  B0 20                     mov al,20h
0E83  FE C6                     inc dh
0E85  E8 04CF R                 call gotoxy
0E88  E8 0FD7 R                 call p_char
0E8B  FE CE                     dec dh
0E8D  FE CE                     dec dh
0E8F  E8 04CF R                 call gotoxy
0E92  E8 0FD7 R                 call p_char
0E95  8B 16 0114 R              mov dx,cursor_position
0E99                    is_not_79:
0E99  FE C2                     inc dl
0E9B  E8 04CF R                 call gotoxy
0E9E  C3                        ret
                        print_english endp
                        ;-------------------------
0E9F                    print_thai proc near
0E9F  50                        push ax
0EA0  50                        push ax
0EA1  A1 0107 R                 mov ax,pointer
0EA4  8B 1E 0111 R              mov bx,col
0EA8  3B D8                     cmp bx,ax
0EAA  58                        pop ax
0EAB  74 19                     je is_overwritet
0EAD  80 3E 0109 R 01           cmp mode_ins,1  ;1 is ins mode
0EB2  75 12                     jne is_overwritet
0EB4  BB 02B4 R                 mov bx,offset ascii_table
0EB7  D7                        xlatb
0EB8  3C D7                     cmp al,215
0EBA  73 0A                     jae is_overwritet
0EBC  80 3E 010F R 01           cmp linestatus,1
0EC1  75 03                     jne is_overwritet
```

```
0EC3  E8 0D92 R                       call shift
0EC6                                  is_overwritet:
0EC6  58                              pop ax
0EC7  50                              push ax
0EC8  E8 04C2 R                       call getxy
0ECB  80 FA 4F                        cmp dl,79
0ECE  75 23                           jne is_conthai
0ED0  8B 1E 0111 R                    mov bx,col
0ED4  83 EB 4E                        sub bx,78
0ED7  8B F3                           mov si,bx
0ED9  E8 04C2 R                       call getxy
0EDC  B2 00                           mov dl,0
0EDE  4E                              dec si
0EDF  B9 004F                         mov cx,79
0EE2  E8 051B R                       call clearline
0EE5  46                              inc si
0EE6  B2 00                           mov dl,0
0EE8  B9 004F                         mov cx,79
0EEB  E8 04DA R                       call printline     ;si is pointer in b
                                      uffer dx is position on screen
0EEE  B2 4E                           mov dl,78    ;set cursor in this posit
                                      ion
0EF0  E8 04CF R                       call gotoxy
0EF3                                  is_conthai:
0EF3  58                              pop ax
0EF4  BB 02B4 R                       mov bx,offset ascii_table
0EF7  D7                              xlatb
0EF8  80 3E 010F R 00                 cmp linestatus,0       ;1 is middle ,
                                      0 is lower and 2 is upper
0EFD  75 24                           jne not_inlower_status
0EFF  3C D7                           cmp al,215             ;is in lower po
                                      sition then check char input
0F01  74 05                           je is_215_216          ;check is lower
                                       character
0F03  3C D8                           cmp al,216
0F05  74 01                           je is_215_216          ;check is lower
                                       character
0F07  C3                              ret
0F08                                  is_215_216:
                                      ;          mov  bx,offset ascii_table     ;subro
                                      utine to write lower character
```

```
                        ;         xlatb              ;at lower charac
                        ter
0F0B  E8 0FD7 R                   call p_char
0F0B  C6 06 0119 R 02             mov choice,2
0F10  A2 0118 R                   mov character,al
0F13  E8 04C2 R                   call getxy          ;put cursor p
                        osition
0F16  FE C2                       inc dl
0F18  FE CE                       dec dh
0F1A  E8 04CF R                   call gotoxy
0F1D  C6 06 010F R 01             mov linestatus,1    ;set linestatus to
                         middle
0F22  C3                          ret
0F23                    not_inlower_status:
0F23  80 3E 010F R 02             cmp linestatus,2    ;check is upper
                        character
0F28  75 28                       jne is_in_middle
0F2A  3C D9                       cmp al,217          ;check is upper c
                        haracter
0F2C  73 01                       jnb is_upper_char
0F2E  C3                          ret
0F2F                    is_upper_char:                ;subroutine to
                         write upper character
                        ;         mov  bx,offset ascii_table   ;at upper
                         position
                        ;         xlatb
0F2F  8B 1E 0111 R                mov bx,col                ;at mid
                        dle position
0F33  8A 9F 011B R                mov bl,line1[bx]
0F37  E8 0FE9 R                   call write_upper
0F3A  A2 0118 R                   mov character,al
0F3D  C6 06 0119 R 03             mov choice,3
0F42  E8 04C2 R                   call getxy          ;put cursor po
                        sition
0F45  FE C2                       inc dl
0F47  FE C6                       inc dh
0F49  E8 04CF R                   call gotoxy
0F4C  C6 06 010F R 01             mov linestatus,1    ;set linestatus to
                        middle
0F51  C3                          ret
0F52                    is_in_middle:                 ;wirte chara
```

```
                        cter at middle position
                        ;       mov  bx,offset ascii_table
                        ;       xlatb
0F52  3C D7                     cmp al,215                      ;check
                         is lower character
0F54  73 03                     jnb isnot_english_mode          ;check
                         is not middle character
0F56  EB 46 90                  jmp is_english_mode
0F59                    isnot_english_mode:
0F59  3C D9                     cmp al,217                      ;check
                         char is lower or upper
0F5B  73 03                     jnb isnot_lower_char            ; is u
                        pper
0F5D  EB 16 90                  jmp is_lower_char               ; is l
                        ower
                        ;*-*-*-*-*-*-*-*-*-*-upper character
0F60                    isnot_lower_char:                       ;subro
                        utine to write upper character
0F60  8B 1E 0111 R              mov bx,col                      ;at mi
                        ddle position
0F64  4B                        dec bx
0F65  8A 9F 011B R              mov bl,line1[bx]
0F69  80 FB D9                  cmp bl,217      ;check  before char is
                         have upper character
0F6C  73 03                     jae is_have_before
0F6E  E9 10CD R                 jmp upper_con1
0F71                    is_have_before:
0F71  E8 0FE9 R                 call write_upper
0F74  C3                        ret
                        ;*-*-*-*-*-*-*-*-*-*-*-*-               lower ch
                        aracter
0F75                    is_lower_char:
0F75  E8 04C2 R                 call getxy
0F78  80 FA 00                  cmp dl,0
0F7B  75 01                     jne lower_con
0F7D  C3                        ret
0F7E                    lower_con:
0F7E  FE C6                     inc dh
0F80  FE CA                     dec dl
0F82  E8 04CF R                 call gotoxy
0F85  E8 0FD7 R                 call p_char
```

```
0F88  E8 04C2 R                         call getxy
0F8B  FE C2                             inc dl
0F8D  E8 04CF R                         call gotoxy
0F90  C6 06 0119 R 05                   mov choice,5
0F95  A2 0118 R                         mov character,al
0F98  FE CE                             dec dh
0F9A  E8 04CF R                         call gotoxy
0F9D  C3                                ret
0F9E                            is_english_mode:
0F9E  E8 0FD7 R                         call p_char         ;is in english mo
                                de
0FA1  A2 0118 R                         mov character,al
0FA4  C6 06 0119 R 01                   mov choice,1
0FA9  B0 20                             mov al,20h
0FAB  FE C6                             inc dh
0FAD  E8 04CF R                         call gotoxy
0FB0  E8 0FD7 R                         call p_char
0FB3  FE CE                             dec dh
0FB5  FE CE                             dec dh
0FB7  E8 04CF R                         call gotoxy
0FBA  E8 0FD7 R                         call p_char
0FBD  FE C6                             inc dh
0FBF  FE C2                             inc dl
0FC1  E8 04CF R                         call gotoxy
0FC4  C3                                ret
                                print_thai endp
                                ;---------------------------
0FC5                            disp_char proc near
0FC5  80 3E 010D R 01                   cmp mode,1          ;1 is thai mode
                                check mode
0FCA  75 03                             jne is_eng_mode
0FCC  EB 05 90                          jmp is_thai_mode    ;is thai mode the
                                n goto is_thai_mode
0FCF                            is_eng_mode:
0FCF  EB 0E05 R                           call print_english
0FD2  C3                                  ret
0FD3                            is_thai_mode:
0FD3  E8 0E9F R                           call print_thai
0FD6  C3                                ret
                                disp_char endp
                                ;---------------------------
```

```
0FD7                    p_char proc near
0FD7  50                       push ax
0FD8  53                       push bx
0FD9  51                       push cx
0FDA  B4 09                    mov ah,09h
0FDC  B7 00                    mov bh,0
0FDE  B3 07                    mov bl,7
0FE0  B9 0001                  mov cx,1
0FE3  CD 10                    int 10h
0FE5  59                       pop cx
0FE6  5B                       pop bx
0FE7  58                       pop ax
0FE8  C3                       ret
                        p_char endp
                        ;-----------------------------
0FE9                    write_upper proc near
0FE9  80 FB DE                 cmp bl,222     ;bl have range 217-222

0FEC  76 03                    jbe is_looktable1
0FEE  EB 52 90                 jmp is_looktable2
0FF1                    is_looktable1:
0FF1  3C E4                    cmp al,228
0FF3  75 0A                    jne is_not_2178
0FF5  80 FB D9                 cmp bl,217
0FF8  75 05                    jne is_not_2178
0FFA  B0 F2                    mov al,242
0FFC  E9 10CD R                jmp upper_con1
0FFF                    is_not_2178:
0FFF  3C E0                    cmp al,224  ;238,243,247,251,234,230

1001  75 0C                    jne is_not_224
1003  B7 00                    mov bh,0
1005  80 EB D9                 sub bl,217
1008  BA B7 02B3 R             mov al,table1[bx]
100C  E9 10CD R                jmp upper_con1
100F                    is_not_224:
100F  3C E1                    cmp al,225     ;239,244,248,252,235,
                        231
1011  75 0C                    jne is_not_225
1013  B7 00                    mov bh,0
1015  80 EB D9                 sub bl,217
```

```
1069  B4 00                         mov ah,0
106B  2D 00E0                       sub ax,224
106E  8B D8                         mov bx,ax
1070  8A B7 029B R                  mov al,table5[bx]
1074  EB 57 90                      jmp upper_con1
1077                          is_more_233:
1077  80 FB ED                      cmp bl,237   ; 230,231,232,233
107A  77 0E                         ja is_more_237
107C  B4 00                         mov ah,0
107E  2D 00E0                       sub ax,224
1081  8B D8                         mov bx,ax
1083  8A B7 029F R                  mov al,table6[bx]
1087  EB 44 90                      jmp upper_con1
108A                          is_more_237:
108A  80 FB F2                      cmp bl,242   ; 230,231,232,233
108D  77 0E                         ja is_more_242
108F  B4 00                         mov ah,0
1091  2D 00E0                       sub ax,224
1094  8B D8                         mov bx,ax
1096  8A B7 02A3 R                  mov al,table7[bx]
109A  EB 31 90                      jmp upper_con1
109D                          is_more_242:
109D  80 FB F6                      cmp bl,246   ; 230,231,232,233
10A0  77 0E                         ja is_more_246
10A2  B4 00                         mov ah,0
10A4  2D 00E0                       sub ax,224
10A7  8B D8                         mov bx,ax
10A9  8A B7 02A8 R                  mov al,table8[bx]
10AD  EB 1E 90                      jmp upper_con1
10B0                          is_more_246:
10B0  80 FB FA                      cmp bl,250
10B3  77 0E                         ja is_more_250
10B5  B4 00                         mov ah,0
10B7  2D 00E0                       sub ax,224
10BA  8B D8                         mov bx,ax
10BC  8A B7 02AC R                  mov al,table9[bx]
10C0  EB 0B 90                      jmp upper_con1
10C3                          is_more_250:
10C3  2C E0                         sub al,224
10C5  B4 00                         mov ah,0
10C7  8B D8                         mov bx,ax
```

```
10C9  8A B7 02B0 R                    mov al,table10[bx]
10CD                            upper_con1:
10CD  80 3E 010F R 01                 cmp linestatus,1
10D2  74 07                           je is_con_upper
10D4  E8 04C2 R                       call getxy
10D7  E8 0FD7 R                       call p_char
10DA  C3                              ret
10DB                            is_con_upper:
10DB  E8 04C2 R                       call getxy
10DE  80 FA 00                        cmp dl,0
10E1  75 01                           jne upper_con2
10E3  C3                              ret
10E4                            upper_con2:
10E4  FE CE                           dec dh
10E6  FE CA                           dec dl
10E8  E8 04CF R                       call gotoxy
10EB  E8 0FD7 R                       call p_char
10EE  E8 04C2 R                       call getxy
10F1  FE C2                           inc dl
10F3  E8 04CF R                       call gotoxy
10F6  C6 06 0119 R 04                 mov choice,4
10FB  A2 0118 R                       mov character,al
10FE  FE C6                           inc dh
1100  E8 04CF R                       call gotoxy
1103  C3                              ret
                                write_upper endp
                                ;-------------------------
1104                            savebuffer proc near
1104  A0 0118 R                       mov al,character
1107  80 3E 0119 R 01                 cmp choice,1
110C  74 03                           je is_1
110E  E9 11E6 R                       jmp is_not1
1111                            is_1:
                                      memory 2          ;save character
                                into middle buffer
1111  53                       1      push bx
1112  50                       1      push ax
1113  B3 02                    1      mov bl,2
1115  80 FB 01                 1      cmp bl,1
1118  75 13                    1      jne ??0003
111A  FF 0E 0111 R             1      dec col
```

```
111E  8B 1E 0111 R          1          mov bx,col
1122  88 87 011B R          1          mov line1[bx],al
1126  FF 06 0111 R          1          inc col
112A  EB 29 90              1          jmp ??0005
112D                        1 ??0003:
112D  80 FB 03              1          cmp bl,3
1130  75 13                 1          jne ??0004
1132  FF 0E 0111 R          1          dec col
1136  8B 1E 0111 R          1          mov bx,col
113A  88 87 020B R          1          mov line3[bx],al
113E  FF 06 0111 R          1          inc col
1142  EB 11 90              1          jmp ??0005
1145                        1 ??0004:
1145  8B 1E 0111 R          1          mov bx,col
1149  88 87 0193 R          1          mov line2[bx],al
114D  FF 06 0107 R          1          inc pointer
1151  FF 06 0111 R          1          inc col
1155                        1 ??0005:
1155  58                    1          pop ax
1156  5B                    1          pop bx
1157  B0 20                              mov al,20h          ;in english mode

                                         memory 1
1159  53                    1          push bx
115A  50                    1          push ax
115B  B3 01                 1          mov bl,1
115D  80 FB 01              1          cmp bl,1
1160  75 13                 1          jne ??0006
1162  FF 0E 0111 R          1          dec col
1166  8B 1E 0111 R          1          mov bx,col
116A  88 87 011B R          1          mov line1[bx],al
116E  FF 06 0111 R          1          inc col
1172  EB 29 90              1          jmp ??0008
1175                        1 ??0006:
1175  80 FB 03              1          cmp bl,3
1178  75 13                 1          jne ??0007
117A  FF 0E 0111 R          1          dec col
117E  8B 1E 0111 R          1          mov bx,col
1182  88 87 020B R          1          mov line3[bx],al
1186  FF 06 0111 R          1          inc col
118A  EB 11 90              1          jmp ??0008
```

```
118D                    1 ??0007:
118D  8B 1E 0111 R      1         mov bx,col
1191  88 87 0193 R      1         mov line2[bx],al
1195  FF 06 0107 R      1         inc pointer
1199  FF 06 0111 R      1         inc col
119D                    1 ??0008:
119D  58                1         pop ax
119E  5B                1         pop bx
                                     memory 3
119F  53                1         push bx
11A0  50                1         push ax
11A1  B3 03             1         mov bl,3
11A3  80 FB 01          1         cmp bl,1
11A6  75 13             1         jne ??0009
11A8  FF 0E 0111 R      1         dec col
11AC  8B 1E 0111 R      1         mov bx,col
11B0  88 87 011B R      1         mov line1[bx],al
11B4  FF 06 0111 R      1         inc col
11B8  EB 29 90          1         jmp ??000B
11BB                    1 ??0009:
11BB  80 FB 03          1         cmp bl,3
11BE  75 13             1         jne ??000A
11C0  FF 0E 0111 R      1         dec col
11C4  8B 1E 0111 R      1         mov bx,col
11C8  88 87 020B R      1         mov line3[bx],al
11CC  FF 06 0111 R      1         inc col
11D0  EB 11 90          1         jmp ??000B
11D3                    1 ??000A:
11D3  8B 1E 0111 R      1         mov bx,col
11D7  88 87 0193 R      1         mov line2[bx],al
11DB  FF 06 0107 R      1         inc pointer
11DF  FF 06 0111 R      1         inc col
11E3                    1 ??000B:
11E3  58                1         pop ax
11E4  5B                1         pop bx
11E5  C3                            ret
11E6                    is_not1:
11E6  80 3E 0119 R 02                 cmp choice,2
11EB  75 4B                           jne is_not2
11ED  FF 06 0111 R                    inc col
                                      memory 3          ;save into lower m
```

```
                                emory
11F1  53                    1           push bx
11F2  50                    1           push ax
11F3  B3 03                 1           mov bl,3
11F5  80 FB 01              1           cmp bl,1
11F8  75 13                 1           jne ??000C
11FA  FF 0E 0111 R          1           dec col
11FE  8B 1E 0111 R          1           mov bx,col
1202  88 87 011B R          1           mov line1[bx],al
1206  FF 06 0111 R          1           inc col
120A  EB 29 90              1           jmp ??000E
120D                        1 ??000C:
120D  80 FB 03              1           cmp bl,3
1210  75 13                 1           jne ??000D
1212  FF 0E 0111 R          1           dec col
1216  8B 1E 0111 R          1           mov bx,col
121A  88 87 020B R          1           mov line3[bx],al
121E  FF 06 0111 R          1           inc col
1222  EB 11 90              1           jmp ??000E
1225                        1 ??000D:
1225  8B 1E 0111 R          1           mov bx,col
1229  88 87 0193 R          1           mov line2[bx],al
122D  FF 06 0107 R          1           inc pointer
1231  FF 06 0111 R          1           inc col
1235                        1 ??000E:
1235  58                    1           pop ax
1236  5B                    1           pop bx
1237  C3                                ret
1238                        is_not2:
1238  80 3E 0119 R 03                     cmp choice,3
123D  75 4B                               jne is_not3
123F  FF 06 0111 R                        inc col
                                          memory 1    ;save character into uppe
                              r character  at upper position
1243  53                    1           push bx
1244  50                    1           push ax
1245  B3 01                 1           mov bl,1
1247  80 FB 01              1           cmp bl,1
124A  75 13                 1           jne ??000F
124C  FF 0E 0111 R          1           dec col
1250  8B 1E 0111 R          1           mov bx,col
```

```
1254  88 87 011B R          1        mov line1[bx],al
1258  FF 06 0111 R          1        inc col
125C  EB 29 90              1        jmp ??0011
125F                        1 ??000F:
125F  80 FB 03              1        cmp bl,3
1262  75 13                 1        jne ??0010
1264  FF 0E 0111 R          1        dec col
1268  8B 1E 0111 R          1        mov bx,col
126C  88 87 020B R          1        mov line3[bx],al
1270  FF 06 0111 R          1        inc col
1274  EB 11 90              1        jmp ??0011
1277                        1 ??0010:
1277  8B 1E 0111 R          1        mov bx,col
127B  88 87 0193 R          1        mov line2[bx],al
127F  FF 06 0107 R          1        inc pointer
1283  FF 06 0111 R          1        inc col
1287                        1 ??0011:
1287  58                   1        pop ax
1288  5B                   1        pop bx
1289  C3                   1          ret
128A                         is_not3:
128A  80 3E 0119 R 04          cmp choice,4
128F  75 47                    jne is_not4
                               memory 1        ;save character int
                        o upper memory
1291  53                   1        push bx
1292  50                   1        push ax
1293  B3 01                1        mov bl,1
1295  80 FB 01             1        cmp bl,1
1298  75 13                1        jne ??0012
129A  FF 0E 0111 R          1        dec col
129E  8B 1E 0111 R          1        mov bx,col
12A2  88 87 011B R          1        mov line1[bx],al
12A6  FF 06 0111 R          1        inc col
12AA  EB 29 90              1        jmp ??0014
12AD                        1 ??0012:
12AD  80 FB 03              1        cmp bl,3
12B0  75 13                 1        jne ??0013
12B2  FF 0E 0111 R          1        dec col
12B6  8B 1E 0111 R          1        mov bx,col
12BA  88 87 020B R          1        mov line3[bx],al
```

```
12BE  FF 06 0111 R          1         inc col
12C2  EB 11 90              1         jmp ??0014
12C5                        1 ??0013:
12C5  8B 1E 0111 R          1         mov bx,col
12C9  88 87 0193 R          1         mov line2[bx],al
12CD  FF 06 0107 R          1         inc pointer
12D1  FF 06 0111 R          1         inc col
12D5                        1 ??0014:
12D5  58                    1         pop ax
12D6  5B                    1         pop bx
12D7  C3                                ret
12D8                          is_not4:
12D8  80 3E 0119 R 05                   cmp choice,5
12DD  75 47                             jne is_not5
                                        memory 3        ;1 is upper,2 is midd
                                  le,3 is upper
12DF  53                    1         push bx
12E0  50                    1         push ax
12E1  B3 03                 1         mov bl,3
12E3  80 FB 01              1         cmp bl,1
12E6  75 13                 1         jne ??0015
12E8  FF 0E 0111 R          1         dec col
12EC  8B 1E 0111 R          1         mov bx,col
12F0  88 87 011B R          1         mov line1[bx],al
12F4  FF 06 0111 R          1         inc col
12F8  EB 29 90              1         jmp ??0017
12FB                        1 ??0015:
12FB  80 FB 03              1         cmp bl,3
12FE  75 13                 1         jne ??0016
1300  FF 0E 0111 R          1         dec col
1304  8B 1E 0111 R          1         mov bx,col
1308  88 87 020B R          1         mov line3[bx],al
130C  FF 06 0111 R          1         inc col
1310  EB 11 90              1         jmp ??0017
1313                        1 ??0016:
1313  8B 1E 0111 R          1         mov bx,col
1317  88 87 0193 R          1         mov line2[bx],al
131B  FF 06 0107 R          1         inc pointer
131F  FF 06 0111 R          1         inc col
1323                        1 ??0017:
1323  58                    1         pop ax
```

```
1324  5B                    1         pop bx
1325  C3                              ret
1326                        is_not5:
                                      memory 2       ;save into middle memo
                            ry buffer
1326  53                    1         push bx
1327  50                    1         push ax
1328  B3 02                 1         mov bl,2
132A  80 FB 01              1         cmp bl,1
132D  75 13                 1         jne ??0018
132F  FF 0E 0111 R          1         dec col
1333  8B 1E 0111 R          1         mov bx,col
1337  88 87 011B R          1         mov line1[bx],al
133B  FF 06 0111 R          1         inc col
133F  EB 29 90              1         jmp ??001A
1342                        1 ??0018:
1342  80 FB 03              1         cmp bl,3
1345  75 13                 1         jne ??0019
1347  FF 0E 0111 R          1         dec col
134B  8B 1E 0111 R          1         mov bx,col
134F  88 87 020B R          1         mov line3[bx],al
1353  FF 06 0111 R          1         inc col
1357  EB 11 90              1         jmp ??001A
135A                        1 ??0019:
135A  8B 1E 0111 R          1         mov bx,col
135E  88 87 0193 R          1         mov line2[bx],al
1362  FF 06 0107 R          1         inc pointer
1366  FF 06 0111 R          1         inc col
136A                        1 ??001A:
136A  58                    1         pop ax
136B  5B                    1         pop bx
136C  C3                              ret
                            savebuffer endp
136D                        skiphead:
136D  E8 0334 R             call init
1370  C6 06 010C R 06              mov direction,6
1375  C6 06 010E R 00              mov noline,0
137A  C6 06 010A R 00              mov y1,0
137F  C6 06 010B R 18              mov y2,24
1384  E8 04A2 R                    call scroll
1387  B6 02                        mov dh,02
```

```
1387  B2 00                     mov dl,00
1389  E8 04CF R                 call gotoxy
138E  8B 1E 0103 R              mov bx,first
1392  89 1E 0116 R              mov line,bx
1396                            begin:
1396  B4 00                     mov ah,0
1398  CD 16                     int 16h
139A  3D 5900                   cmp ax,5900h
139D  73 03                     jnb functionkey
139F  EB 03 90                  jmp not_functionkey
13A2                            functionkey:
13A2  EB F2                      jmp begin
13A4                            not_functionkey:
13A4  3D 1000                   cmp ax,1000h            ;alt-q is exit
13A7  75 03                     jne not_exit
13A9  E9 1430 R                 jmp exit
13AC                            not_exit:
13AC  3D 4B00                   cmp ax,4b00h            ;left
13AF  75 05                     jne not_left
13B1  E8 0554 R                 call left
13B4  EB E0                     jmp begin
13B6                            not_left:
13B6  3D 4D00                   cmp ax,4d00h            ;right
13B9  75 05                     jne not_right
13BB  E8 0613 R                 call right
13BE  EB D6                     jmp begin
13C0                            not_right:
13C0  3D 4800                   cmp ax,4800h            ;up
13C3  75 05                     jne not_up
13C5  E8 0A2E R                 call up
13C8  EB CC                     jmp begin
13CA                            not_up:
13CA  3L 5000                   cmp ax,5000h            ;down
13CD  75 05                     jne not_down
13CF  E8 0B11 R                 call down
13D2  EB C2                     jmp begin
13D4                            not_down:
13D4  3C 08                     cmp al,08               ; rubout
13D6  75 05                     jne not_rubout
13D8  E8 0BF1 R                 call rubout
13DB  EB B9                     jmp begin
```

```
13DD                    not_rubout:
13DD  3D 5300                   cmp ax,5300h            ;delete
13E0  75 05                     jne not_delete
13E2  E8 0019 R                 call delete
13E5  EB AF                     jmp begin
13E7                    not_delete:
13E7  3D 5200                   cmp ax,5200h
13EA  75 06                     jne not_ins
13EC  F6 16 010D R              not mode_ins
13F0  EB 14                     jmp begin
13F2                    not_ins:
13F2  3D 4900                   cmp ax,4900h            ;pageup
13F5  75 02                     jne not_pageup
                        ;       call pageup
13F7  EB 9D                     jmp begin
13F9                    not_pageup:
13F9  3D 5100                   cmp ax,5100h            ;pagedown
13FC  75 02                     jne not_pagedown
                        ;       call pagedown
13FE  EB 9A                     jmp begin
1400                    not_pagedown:
1400  3C 0D                     cmp al,13               ;return
1402  75 05                     jne not_return
1404  E8 0785 R                 call return
1407  EB 8D                     jmp begin
1409                    not_return:
1409  3C 1B                     cmp al,27               ;esc is change mode
140B  75 06                     jne not_esc
140D  F6 16 010D R              not mode
1411  EB 83                     jmp begin
1413                    not_esc:
1413  83 3E 0111 R 77           cmp col,119             ;check end of line
1418  76 03                     jbe is_less_119
141A  E9 1396 R                 jmp begin
141D                    is_less_119:
141D  80 3E 011A R A0           cmp counter_line,160    ;check counter of
                        line 0-159
1422  76 03                     jbe is_less_160         ;******* will cha
                        nge
1424  E9 1396 R                 jmp begin
1427                    is_less_160:
```

```
1427  E8 0FC5 R                 call disp_char
142A  E8 1104 R                 call savebuffer
142D  E9 1396 R                 jmp begin
1430                      exit:
1430  CD 20                     int 20h
1432                      seg_end:
1432                      code ends
                          end start
```

Microsoft (R) Macro Assembler Version 4.00 1/1/80 00:36:17

project Symbols-1

Macros:

Name	Lines
MEMORY	26

Segments and Groups:

Name	Size	Align	Combine Class
CODE	1432	PARA	NONE

Symbols:

Name	Type	Value	Attr	
AGAIN_CLEAR	L NEAR	0522	CODE	
AGAIN_DELETE	L NEAR	0D71	CODE	
AGAIN_PRINT	L NEAR	04E2	CODE	
AGAIN_RUB	L NEAR	0C57	CODE	
AGAIN_SHIFT	L NEAR	0DA7	CODE	
ASCII_TABLE	L BYTE	02B4	CODE	Length = 0020
BEFORE_CHAR	L BYTE	0110	CODE	
BEGIN	L NEAR	1396	CODE	
BEGINDATA	L NEAR	06E9	CODE	
BEGINRUB	L NEAR	0CAA	CODE	
BEGINRUB1	L NEAR	0CF3	CODE	
BEGIN_ENG	L NEAR	0E2E	CODE	
CHARACTER	L BYTE	0118	CODE	
CHAR_EXIT	L NEAR	047C	CODE	
CHAR_LINE	N PROC	046D	CODE	Length = 0011
CHOICE	L BYTE	0119	CODE	
CLEAR	L NEAR	09FC	CODE	
CLEARBUFFER	N PROC	0727	CODE	Length = 0015
CLEARLINE	N PROC	051B	CODE	Length = 0039
CLEAR_APPEND	L NEAR	0913	CODE	
CNGLENG	N PROC	0388	CODE	Length = 000D
COL	L WORD	0111	CODE	
COUNTER_LINE	L BYTE	011A	CODE	

CR_CON	L NEAR	0891	CODE	
CR_CON1	L NEAR	0973	CODE	
CURSOR_POSITION	L WORD	0114	CODE	
DECPRINT	N PROC	047E	CODE	Length = 0024
DECPRINT1	L NEAR	0489	CODE	
DECPRINT2	L NEAR	0493	CODE	
DELETE	N PROC	0D19	CODE	Length = 0079
DELLINE	N PROC	03DC	CODE	Length = 0023
DIRECTION	L BYTE	010C	CODE	
DISP_CHAR	N PROC	0FC5	CODE	Length = 0012
DOWN	N PROC	0B11	CODE	Length = 00E0
DSIS_MIDDLE	L NEAR	0D47	CODE	
DSNOT_MIDDLE	L NEAR	0D23	CODE	
DSNOT_UP	L NEAR	0D3A	CODE	
ERRPROC	N PROC	037A	CODE	Length = 0001
EXIT	L NEAR	1430	CODE	
EXIT1	L NEAR	0D18	CODE	
EXITRUBOUT	L NEAR	0CDC	CODE	
EXIT_LOOP	L NEAR	0DDA	CODE	
FIRST	L WORD	0103	CODE	
FUNCTIONKEY	L NEAR	13A2	CODE	
GETLENG	N PROC	037B	CODE	Length = 000D
GETXY	N PROC	04C2	CODE	Length = 000D
GOTOXY	N PROC	04CF	CODE	Length = 000B
INIT	N PROC	0334	CODE	Length = 002F
INSBEFORE	N PROC	03FF	CODE	Length = 0045
INSBEXIT	L NEAR	043E	CODE	
INSB_1	L NEAR	0411	CODE	
INSB_2	L NEAR	0436	CODE	
INSB_3	L NEAR	043C	CODE	
INSLINE	N PROC	0395	CODE	Length = 0047
INS_1	L NEAR	03A7	CODE	
INS_2	L NEAR	03CE	CODE	
INS_3	L NEAR	03D4	CODE	
INS_EXT	L NEAR	03D6	CODE	
IS239_242	L NEAR	1061	CODE	

ISNOT_ENGLISH_MODE	L NEAR	0F59	CODE
ISNOT_LOWER	L NEAR	0CCA	CODE
ISNOT_LOWER_CHAR	L NEAR	0F60	CODE
ISNOT_UPPER	L NEAR	0C8F	CODE
ISN_LOWER	L NEAR	0D10	CODE
ISN_UPPER	L NEAR	0D05	CODE
ISN_Y20	L NEAR	0878	CODE
ISN_Y_23	L NEAR	086F	CODE
ISRE_OVERWRITE	L NEAR	07BE	CODE
IS_1	L NEAR	1111	CODE
IS_215_216	L NEAR	0F08	CODE
IS_CANNOT	L NEAR	0B0F	CODE
IS_CANNOT1	L NEAR	0BEF	CODE
IS_CON	L NEAR	061B	CODE
IS_CON1	L NEAR	0E6E	CODE
IS_CON11	L NEAR	055C	CODE
IS_CON1111	L NEAR	0AA5	CODE
IS_CON2222	L NEAR	0B0B	CODE
IS_CON3333	L NEAR	0B8B	CODE
IS_CON4444	L NEAR	0BEB	CODE
IS_CONR	L NEAR	0625	CODE
IS_CONR1	L NEAR	09AE	CODE
IS_CONRI	L NEAR	09BE	CODE
IS_CONTHAI	L NEAR	0EF3	CODE
IS_CON_DEL	L NEAR	0D53	CODE
IS_CON_UPPER	L NEAR	10DB	CODE
IS_CX_ZERO	L NEAR	083C	CODE
IS_ENGLISH_MODE	L NEAR	0F9E	CODE
IS_ENG_MODE	L NEAR	0FCF	CODE
IS_EXIT_CR	L NEAR	09C1	CODE
IS_FIRST2	L NEAR	0BA0	CODE
IS_HAVE_BEFORE	L NEAR	0F71	CODE
IS_HAVE_LAST	L NEAR	07DE	CODE
IS_IN_MIDDLE	L NEAR	0F52	CODE
IS_LESS79	L NEAR	0DAF	CODE
IS_LESS80	L NEAR	0835	CODE
IS_LESS_119	L NEAR	141D	CODE
IS_LESS_160	L NEAR	1427	CODE
IS_LESS_228	L NEAR	1064	CODE
IS_LOOKTABLE1	L NEAR	0FF1	CODE
IS_LOOKTABLE2	L NEAR	1042	CODE

IS_LOWER_CHAR	L NEAR	0F75	CODE
IS_L_NOTUPPER	L NEAR	0573	CODE
IS_MIDDLELOWER	L NEAR	0779	CODE
IS_MORE_223	L NEAR	1051	CODE
IS_MORE_230	L NEAR	104A	CODE
IS_MORE_233	L NEAR	1077	CODE
IS_MORE_237	L NEAR	108A	CODE
IS_MORE_242	L NEAR	109D	CODE
IS_MORE_246	L NEAR	10B0	CODE
IS_MORE_250	L NEAR	10C3	CODE
IS_NOHAVEUPPER	L NEAR	04FE	CODE
IS_NOLOWER	L NEAR	0510	CODE
IS_NOT1	L NEAR	11E6	CODE
IS_NOT2	L NEAR	1238	CODE
IS_NOT3	L NEAR	128A	CODE
IS_NOT4	L NEAR	12D8	CODE
IS_NOT5	L NEAR	1326	CODE
IS_NOTCR	L NEAR	075F	CODE
IS_NOT_2178	L NEAR	0FFF	CODE
IS_NOT_224	L NEAR	100F	CODE
IS_NOT_225	L NEAR	101F	CODE
IS_NOT_226	L NEAR	102F	CODE
IS_NOT_227	L NEAR	103F	CODE
IS_NOT_79	L NEAR	0E99	CODE
IS_NOT_BEGIN	L NEAR	0C27	CODE
IS_NOT_CR	L NEAR	06FC	CODE
IS_NOT_END	L NEAR	0C3D	CODE
IS_NOT_FIRST	L NEAR	0A84	CODE
IS_NOT_FIRST1	L NEAR	0AC0	CODE
IS_NOT_FIRST2	L NEAR	0B67	CODE
IS_NOT_LOWERDATA	L NEAR	0724	CODE
IS_NOT_OUT	L NEAR	0CB4	CODE
IS_NO_LOWER	L NEAR	054A	CODE
IS_NO_UPPER	L NEAR	0536	CODE
IS_NO_UPPERDATA	L NEAR	0714	CODE
IS_N_E_UP	L NEAR	0E1D	CODE
IS_OVERWRITE	L NEAR	0E43	CODE
IS_OVERWRITET	L NEAR	0EC6	CODE
IS_RETURNOK	L NEAR	084D	CODE
IS_RUB_CON	L NEAR	0C9E	CODE
IS_R_NOTUPPER	L NEAR	0643	CODE

IS_THAI_MODE	L NEAR	0FD3	CODE	
IS_UPPER	L NEAR	0770	CODE	
IS_UPPER_CHAR	L NEAR	0F2F	CODE	
LAST	L WORD	0105	CODE	
LCHECK_LOW	L NEAR	05E9	CODE	
LEFT	N PROC	0554	CODE	Length = 00BF
LINE	L WORD	0116	CODE	
LINE1	L BYTE	011B	CODE	Length = 0078
LINE2	L BYTE	0193	CODE	Length = 0078
LINE3	L BYTE	020B	CODE	Length = 0078
LINESTATUS	L BYTE	010F	CODE	
LIS_MIDDLE	L NEAR	05CF	CODE	
LNOT_MIDDLE	L NEAR	05B7	CODE	
LNOT_UP	L NEAR	05CC	CODE	
LOWER_CON	L NEAR	0F7E	CODE	
L_CON	L NEAR	057F	CODE	
MALLOC	N PROC	0363	CODE	Length = 000E
MAXLINE	Number	00A0		
MODE	L BYTE	010D	CODE	
MODE_INS	L BYTE	0109	CODE	
NOLINE	L BYTE	010E	CODE	
NOT_DELETE	L NEAR	13E7	CODE	
NOT_DOWN	L NEAR	13D4	CODE	
NOT_ESC	L NEAR	1413	CODE	
NOT_EXIT	L NEAR	13AC	CODE	
NOT_FUNCTIONKEY	L NEAR	13A4	CODE	
NOT_INLOWER_STATUS	L NEAR	0F23	CODE	
NOT_INS	L NEAR	13F2	CODE	
NOT_LEFT	L NEAR	13B6	CODE	
NOT_PAGEDOWN	L NEAR	1400	CODE	
NOT_PAGEUP	L NEAR	13F9	CODE	
NOT_RETURN	L NEAR	1409	CODE	
NOT_RIGHT	L NEAR	13C0	CODE	
NOT_RUBOUT	L NEAR	13DD	CODE	
NOT_UP	L NEAR	13CA	CODE	
NOT_WRITE_LINE	L NEAR	05AD	CODE	
NXTLINE	N PROC	0444	CODE	Length = 0015
NXTLINE_1	L NEAR	0456	CODE	

```
ONETOTHREE . . . . . .  N PROC  073C   CODE   Length = 0049

POINTER . . . . . .  L WORD  0107   CODE
PRELINE . . . . . .  N PROC  0459   CODE   Length = 0014
PRELINE_1 . . . . . .  L NEAR  046A   CODE
PRINTLINE . . . . . .  N PROC  04DA   CODE   Length = 0041
PRINT_ENGLISH . . . . . .  N PROC  0E05   CODE   Length = 009A
PRINT_THAI . . . . . .  N PROC  0E9F   CODE   Length = 0126
PUT_AGAIN . . . . . .  L NEAR  0753   CODE
PUZZLE . . . . . .  L NEAR  0608   CODE
P_CHAR . . . . . .  N PROC  0FD7   CODE   Length = 0012

RCHECK_LOW . . . . . .  L NEAR  06B7   CODE
RELEASE . . . . . .  N PROC  0371   CODE   Length = 0009
RENOT_MIDDLE . . . . . .  L NEAR  07BF   CODE
RENOT_UP . . . . . .  L NEAR  07A6   CODE
RETURN . . . . . .  N PROC  0785   CODE   Length = 00BC
RETURNINS . . . . . .  N PROC  0841   CODE   Length = 01ED
RE_LOOP . . . . . .  L NEAR  09DD   CODE
RE_MIDDLE . . . . . .  L NEAR  07B3   CODE
RIGHT . . . . . .  N PROC  0613   CODE   Length = 00CE
RISN_Y20 . . . . . .  L NEAR  095A   CODE
RISN_Y_23 . . . . . .  L NEAR  0951   CODE
RIS_MIDDLE . . . . . .  L NEAR  069D   CODE
RMOR23 . . . . . .  L NEAR  0807   CODE
RNOT_MIDDLE . . . . . .  L NEAR  0685   CODE
RNOT_UP . . . . . .  L NEAR  069A   CODE
RPUZZLE . . . . . .  L NEAR  06D6   CODE
RUBNOT_MIDDLE . . . . . .  L NEAR  0BFB   CODE
RUBNOT_UP . . . . . .  L NEAR  0C12   CODE
RUBOUT . . . . . .  N PROC  0BF1   CODE   Length = 012B
RUBOUT_BEGIN . . . . . .  N PROC  0C7C   CODE   Length = 009D
RUB_MIDDLE . . . . . .  L NEAR  0C1F   CODE
R_CON . . . . . .  L NEAR  064F   CODE
R_NOT_EXIT . . . . . .  L NEAR  067B   CODE
R_T . . . . . .  L NEAR  0905   CODE

SAVEBUFFER . . . . . .  N PROC  1104   CODE   Length = 0269
SAVE_ROT . . . . . .  L NEAR  0992   CODE
SCROLL . . . . . .  N PROC  04A2   CODE   Length = 0020
```

SEG_BEGIN	L NEAR	0000	CODE	
SEG_END	L NEAR	1432	CODE	
SHIFT	N PROC	0D92	CODE	Length = 0073
SHIFT1	L NEAR	0DE4	CODE	
SKIPHEAD	L NEAR	136D	CODE	
START	L NEAR	0100	CODE	
TABLE1	L BYTE	0283	CODE	
TABLE10	L BYTE	02B0	CODE	
TABLE2	L BYTE	0289	CODE	
TABLE3	L BYTE	028F	CODE	
TABLE4	L BYTE	0295	CODE	
TABLE5	L BYTE	029B	CODE	
TABLE6	L BYTE	029F	CODE	
TABLE7	L BYTE	02A3	CODE	
TABLE8	L BYTE	02A8	CODE	
TABLE9	L BYTE	02AC	CODE	
THREETOONE	N PROC	06E1	CODE	Length = 0046
TYPE	L BYTE	0113	CODE	
UIS_MIDDLE	L NEAR	0A5C	CODE	
UIS_MIDDLE2	L NEAR	0B3F	CODE	
UNOT_MIDDLE	L NEAR	0A38	CODE	
UNOT_MIDDLE2	L NEAR	0B1B	CODE	
UNOT_UP	L NEAR	0A4F	CODE	
UNOT_UP2	L NEAR	0B32	CODE	
UP	N PROC	0A2E	CODE	Length = 00E3
UPPER_CON1	L NEAR	10CD	CODE	
UPPER_CON2	L NEAR	10E4	CODE	
U_PROBLEM	L NEAR	0AAF	CODE	
U_PROBLEM2	L NEAR	0B8F	CODE	
WRITE_UPPER	N PROC	0FE9	CODE	Length = 011B
Y1	L BYTE	010A	CODE	
Y2	L BYTE	010B	CODE	
??0000	L NEAR	08B4	CODE	
??0001	L NEAR	08CC	CODE	
??0002	L NEAR	08DC	CODE	
??0003	L NEAR	112D	CODE	

??0004	L NEAR	1145	CODE
??0005	L NEAR	1155	CODE
??0006	L NEAR	1175	CODE
??0007	L NEAR	118D	CODE
??0008	L NEAR	119D	CODE
??0009	L NEAR	118B	CODE
??000A	L NEAR	11D3	CODE
??000B	L NEAR	11E3	CODE
??000C	L NEAR	120D	CODE
??000D	L NEAR	1225	CODE
??000E	L NEAR	1235	CODE
??000F	L NEAR	125F	CODE
??0010	L NEAR	1277	CODE
??0011	L NEAR	1287	CODE
??0012	L NEAR	12AD	CODE
??0013	L NEAR	12C5	CODE
??0014	L NEAR	12D5	CODE
??0015	L NEAR	12FB	CODE
??0016	L NEAR	1313	CODE
??0017	L NEAR	1323	CODE
??0018	L NEAR	1342	CODE
??0019	L NEAR	135A	CODE
??001A	L NEAR	136A	CODE

1894 Source Lines
2128 Total Lines
298 Symbols

40022 Bytes symbol space free

0 Warning Errors
0 Severe Errors

```
                        title project
                        page 45,80
= 00A0                  MaxLine Equ 160 ; Max char on a lines
0000                    code segment
                        assume cs:code ,ds:code
0000                            org 0h
0000                    seg_begin:
002C                            org 2ch
002C                    env_seg label word
                        ; DATA TRANFER AREA USE FOR DIRECTORY
0080                            org 80h
0080                    dta     label byte
009E                            org 9eh
009E                    file_found label byte
0100                            org 100h
0100  E9 0B3D R         start:  jmp skiphead
                        ;====================== data area ===============
                        =======
                        ; Data For Link List
0103  ????               First  dw ?
0105  ????               Last   dw ?
0107  ????               Row    dw ?
0109  ????                Col dw ?
010B  02                x db 2
                         ;<-------------  Data For Shell--------------
                        -->
010C  0D 0A 44 4F 53 20 43       Dos_prompt db 0dh,0ah,'DOS COMMAND => $'
      4F 4D 4D 41 4E 44 20
      3D 3E 20 24
                        ;       < Command Line >
011E  20 2F                     cmd_tail db ' /'
0120  1E                        com_buff db 30
0121  ??                                 db ?
0122  0031[                              db 31h dup (0)
          00
                    ]

                        ;       < Comand Line >
0153  ????              stackptr dw ?
0155  ????              stackSEg dw ?
0157  63 6F 6D 6D 61 6E 64     comma db 'command.com',0
```

```
      2E 63 6F 6D 00
                              ;       < Parameter Block >
 0163  0000                   parameter dw 0
 0165  011E R                         dw offset cmd_tail
 0167  0000                   c1       dw 0 ; segment cmd_tail
 0169  FF FF FF FF                     dd -1 ;fcb1
 016D  FF FF FF FF                     dd -1 ; fcb2
                              ;       < Parameter Block >
                              ;<-------------- Screen data ------------------>

 0171  ????                   screen          dw      ?
 0173  00                     mode_sc         db      0
 0174  54 68 72 65 65 20 4C  get_line_mess    db      'Three Line Error Position $'

       69 6E 65 20 45 72 72
       6F 72 20 50 6F 73 69
       74 69 6F 6E 20 24
                              ;<-------------- Disk Data ------------------->

 018F  ????                   handle          dw      ?
 0191  1E                     File_name       db      30
 0192  ??                                     db      ?
 0193  61 62 63 2E 64 65 66   FileName        db      'abc.def',0,20 dup(' ')
       00
       0014[
            20
                         ]

 01AF  000D[                  FILE_1  DB      13 DUP (' ')
            20
                         ]

 01BC  000D[                  FILE_2  DB      13 DUP (' ')
            20
                         ]

 01C9  000D[                  FILE_3  DB      13 DUP (' ')
            20
                         ]

 01D6  000D[                  FILE_4  DB      13 DUP (' ')
```

```
          20
                        ]

01E3  000D[                     FILE_5  DB      13 DUP (' ')
          20
                        ]

01F0  000D[                     FILE_6  DB      13 DUP (' ')
          20
                        ]

01FD  000D[                     FILE_7  DB      13 DUP (' ')
          20
                        ]

020A  000D[                     FILE_8  DB      13 DUP (' ')
          20
                        ]

0217  43 41 4E 27 54 20 46     Print_dir_1 db "CAN'T FIND C:\COMMAND.COM ",0DH,
                               0AH ,'$'
      49 4E 44 20 43 3A 5C
      43 4F 4D 4D 41 4E 44
      2E 43 4F 4D 20 0D 0A
      24

                               ;<-------------- Code Area ------------------->

                               ;====================== Screen control ==========
                               =============
0234                           getxy proc near
0234  50                              push ax
0235  53                              push bx
0236  51                              push cx
0237  B4 03                           mov ah,03
0239  B7 00                           mov bh,0
023B  CD 10                           int 10h
023D  59                              pop cx
023E  5B                              pop bx
023F  58                              pop ax
0240  C3                              ret
                               getxy endp
```

```
0241                            cls proc near
0241  50                                push ax
0242  53                                push bx
0243  51                                push cx
0244  52                                push dx
0245  B6 18                            mov dh,24
0247  B2 4F                            mov dl,79
0249  B9 0000                          mov cx,0
024C  B7 07                            mov bh,07
024E  B0 18                            mov al,24
0250  B4 06                            mov ah,6
0252  CD 10                            int 10h
0254  5A                              pop dx
0255  59                              pop cx
0256  5B                              pop bx
0257  58                              pop ax
0258  C3                              ret
                                cls endp

0259                            gotoxy proc near
0259  50                              push ax
025A  53                              push bx
025B  B4 02                           mov ah,02
025D  B7 00                           mov bh,0
025F  CD 10                           int 10h
0261  5B                              pop bx
0262  58                              pop ax
0263  C3                            ret
                                gotoxy endp
                                ; dh,dl = x,y position in three linee
                                ; dh [0..7],dl,[0..79]
0264                            ToLine proc near
0264  50                              push ax
0265  52                              push dx
0266  B8 0003                        mov ax,03h
0269  F6 E6                          mul dh
026B  04 02                          add al,2
026D  8A F0                          mov dh,al
026F  E8 0259 R                      call gotoxy
0272  5A                              pop dx
0273  58                              pop ax
0274  C3                            ret
```

```
                        ToLine endp
                        getline Proc near
0275                            push ax
0275  50                        push cx
0276  51                        xor ax,ax
0277  33 C0                     call getxy
0279  E8 0234 R                  or dh,dh
027C  0A F6                     jz get_line2
027E  74 16                     dec dh
0280  FE CE                     jnz get_li
0282  75 05                     mov dh,0
0284  B6 00                     jmp get_line1
0286  EB 0B 90                 get_li:
0289                            dec dh
0289  FE CE                     mov al,dh
028B  8A C6                     mov cl,3
028D  B1 03                     div cl
028F  F6 F1                     mov dh,al
0291  8A F0                     get_line1:
0293                            pop cx
0293  59                        pop ax
0294  58                        ret
0295  C3                      get_line2:
0296                            push dx
0296  52                        mov dx,offset get_line_mess
0297  BA 0174 R                 mov ah,9
029A  B4 09                     int 21h
029C  CD 21                     jmp get_line1
029E  EB F3                   getline endp
                              set_screen proc near
02A0                            push ax
02A0  50                        push es
02A1  06                        push di
02A2  57                        mov ax,0b000h
02A3  B8 B000                   mov es,ax
02A6  8E C0                     xor di,di
02A8  33 FF                     mov ah,0feh
02AA  B4 FE                     int 10h
02AC  CD 10                     mov screen,es
02AE  8C 06 0171 R              mov ah,0fh
02B2  B4 0F                     int 10h
02B4  CD 10
```

```
02B6  3C 07                     cmp al,7
02B8  74 04                     jz set_sc1
02BA  F6 16 0173 R              Not mode_sc
02BE                    set_sc1:
02BE  5F                        pop di
02BF  07                        pop es
02C0  5B                        pop ax
02C1  C3                        ret
                        set_screen endp
02C2                    write_direct proc near
02C2  50                        push ax
02C3  53                        push bx
02C4  52                        push dx
02C5  57                        push di
02C6  06                        push es
02C7  8A D8                     mov bl,al
02C9  B0 A0                     mov al,160
02CB  F6 E6                     mul dh
02CD  B6 00                     mov dh,0
02CF  D1 E2                     shl dx,1
02D1  03 C2                     add ax,dx
02D3  8B F8                     mov di,ax
02D5  8E 06 0171 R              mov es,screen
02D9  80 3E 0173 R 00           cmp mode_sc,0
02DE  74 0E                     jz write_di1
02E0  BA 03DA                   mov dx,03dah
02E3  FA                        cli
02E4                    write_wait1:
02E4  EC                        in al,dx
02E5  24 01                     and al,1
02E7  75 FB                     jnz write_wait1
02E9                    write_wait2:
02E9  EC                        in al,dx
02EA  24 01                     and al,1
02EC  74 FB                     jz write_wait2
02EE                    write_di1:
02EE  26: 88 1D                 mov es:[di],bl
02F1  FB                        sti
02F2  07                        pop es
02F3  5F                        pop di
02F4  5A                        pop dx
```

```
02F5  5B                        pop bx
02F6  58                        pop ax
02F7  C3                        ret
                        write_direct endp
02F8                    Println Proc near
02F8  50                        push ax
02F9  53                        push bx
02FA  51                        push cx
02FB  52                        push dx
02FC  56                        push si
02FD  57                        push di
02FE  1E                        push ds
02FF  06                        push es
0300  E8 0275 R                 call GetLine
0303  32 D2                     xor dl,dl       ;write to begin of line
0305  B8 0003                   mov ax,03h      ;cal acture line of middle line
0308  F6 E6                     mul dh
030A  04 02                     add al,2
030C  8A F0                     mov dh,al
030E  B1 00                     mov cl,0
0310  E8 05DC R                 call Char_line ;es:[di] point to data on line
0313  E8 04D7 R                 call GetLeng   ;get line leng
0316  8A 0E 010B R              mov cl,x
031A  3A CD                     cmp cl,ch
031C  77 3E                     ja prln2
031E  FE C1                     inc cl
0320  B7 00                     mov bh,0
0322                    prln00:
0322  26: 8A 05                 mov al,es:[di]
0325  47                        inc di
0326  FE C7                     inc bh
0328  3C D7                     cmp al,215;find middle char
032A  73 F6                     jnb prln00
032C  FE C9                     dec cl
032E  75 F2                     jnz prln00
0330  4F                        dec di
0331  2A EF                     sub ch,bh
0333  74 27                     jz prln2
0335                    prln0:
0335  26: 8A 05                 mov al,es:[di]
0338  47                        inc di
```

```
0339  8A DE                    mov bl,dh
033B  3C D7                    cmp al,215
033D  72 0B                    jb prln1
033F  FE CE                    dec dh
0341  FE CA                    dec dl
0343  3C D9                    cmp al,217
0345  73 03                    jae prln1
0347  80 C6 02                 add dh,2
034A                        prln1:
034A  E8 02C2 R                call write_direct
034D  8A F3                    mov dh,bl
034F  FE C2                    inc dl
0351  80 FA 50                 cmp dl,80          ;end of screen ?
0354  74 06                    jz prln2
0356  FE CD                    dec ch             ;end of data   ?
0358  74 02                    jz prln2
035A  EB D9                    jmp prln0
035C                         prln2:
035C  07                       pop es
035D  1F                       pop ds
035E  5F                       pop di
035F  5E                       pop si
0360  5A                       pop dx
0361  59                       pop cx
0362  5B                       pop bx
0363  58                       pop ax
0364  C3                       ret
                            Println endp
                            ; bx point to cursor point line
0365  ????                  bx_save dw ?
0367                        page_up proc near
0367  50                       push ax
0368  51                       push cx
0369  52                       push dx
036A  56                       push si
036B  57                       push di
036C  1E                       push ds
036D  06                       push es
036E  E8 0241 R                call cls
0371  E8 0275 R                call GetLine
0374  52                       push dx
```

```
0375  B9 0007                    mov cx,7
0378  02 CE                     add cl,dh
037A                          page_up1:
037A  E8 05CA R                 call PreLine
037D  74 02                     jz page_up2
037F  E2 F9                     loop page_up1
0381                          page_up2:
0381  2E: 89 1E 0365 R          mov cs:bx_save,bx
0386  BA 0000                   mov dx,0
0389  B9 000B                   mov cx,8
038C                          page_up3:
038C  0A F6                     or dh,dh
038E  74 05                     jz page_up4
0390  B2 00                     mov dl,0
0392  E8 05B7 R                 call NxtLine
0395                          page_up4:
0395  E8 0264 R                 call toLine
0398  E8 02F8 R                 call printLn
039B  FE C6                     inc dh
039D  74 02                     je page_up5
039F  E2 EB                     loop page_up3
03A1                          page_up5:
03A1  5A                        pop dx
03A2  E8 0264 R                 call toLine
03A5  2E: 8B 1E 0365 R          mov bx,cs:bx_save
03AA  0A F6                     or dh,dh
03AC  74 09                     jz page_up7
03AE  8A CE                     mov cl,dh
03B0  B5 00                     mov ch,0
03B2                          page_up6:
03B2  E8 05B7 R                 call NxtLine
03B5  E2 FB                     loop page_up6
03B7                          page_up7:
03B7  07                        pop es
03B8  1F                        pop ds
03B9  5F                        pop di
03BA  5E                        pop si
03BB  5A                        pop dx
03BC  59                        pop cx
03BD  58                        pop ax
03BE  C3                        ret
```

```
                        page_up endp
 03BF                   page_down proc near
 03BF  50                       push ax
 03C0  51                       push cx
 03C1  52                       push dx
 03C2  56                       push si
 03C3  57                       push di
 03C4  1E                       push ds
 03C5  06                       push es
 03C6  E8 0241 R                call cls
 03C9  E8 0275 R                call GetLine
 03CC  52                       push dx
 03CD  B9 0007                  mov cx,7
 03D0  2A CE                    sub cl,dh
 03D2                   page_down1:
 03D2  E8 05B7 R                call NxtLine
 03D5  74 02                    jz page_down2
 03D7  E2 F9                    loop page_down1
 03D9                   page_down2:
 03D9  2E: 89 1E 0365 R         mov cs:bx_save,bx
 03DE  BA 0000                  mov dx,0
 03E1  B9 0008                  mov cx,8
 03E4                   page_down3:
 03E4  0A F6                    or dh,dh
 03E6  74 07                    jz page_down4
 03E8  B2 00                    mov dl,0
 03EA  E8 0587 R                call NxtLine
 03ED  74 2A                    jz page_down8
 03EF                   page_down4:
 03EF  E8 0264 R                call toLine
 03F2  E8 02F8 R                call println
 03F5  FE C6                    inc dh
 03F7  74 02                    je page_down5
 03F9  E2 E9                    loop page_down3
 03FB                   page_down5:
 03FB  5A                       pop dx
 03FC  E8 0264 R                call toLine
 03FF  2E: 8B 1E 0365 R         mov bx,cs:bx_save
 0404  0A F6                    or dh,dh
 0406  74 09                    jz page_down7
 0408  8A CE                    mov cl,dh
```

```
040A  B5 00                     mov ch,0
040C                        page_down6:
040C  E8 05B7 R                 call NxtLine
040F  E2 FB                     loop page_down6
0411                        page_down7:
0411  07                        pop es
0412  1F                        pop ds
0413  5F                        pop di
0414  5E                        pop si
0415  5A                        pop dx
0416  59                        pop cx
0417  58                        pop ax
0418  C3                        ret
0419                        page_down8:
0419  5A                        pop dx
041A  2E: 8B 1E 0365 R          mov bx,cs:bx_save
041F  BA 0000                   mov dx,0
0422  E8 0264 R                 call toLine
0425  EB EA                     jmp page_down7
                            page_down endp
                            ;input bx point to cursor point position
0427                        page_up_date proc near
0427  50                        push ax
0428  51                        push cx
0429  52                        push dx
042A  56                        push si
042B  57                        push di
042C  1E                        push ds
042D  06                        push es
042E  E8 0241 R                 call cls
0431  E8 0275 R                 call GetLine
0434  52                        push dx
0435  8A CE                     mov cl,dh
0437  B5 00                     mov ch,0
0439  0A F6                     or dh,dh
043B  74 07                     jz page_up_date2
043D                        page_up_date1:
043D  E8 05CA R                 call PreLine
0440  74 02                     jz page_up_date2
0442  E2 F9                     loop page_up_date1
0444                        page_up_date2:
```

```
0444  2E: 89 1E 0365 R          mov cs:bx_save,bx
0449  BA 0000                   mov dx,0
044C  B9 0008                   mov cx,8
044F                        page_up_date3:
044F  0A F6                     or dh,dh
0451  74 05                     jz page_up_date4
0453  B2 00                     mov dl,0
0455  E8 05B7 R                 call NxtLine
0458                        page_up_date4:
0458  E8 0264 R                 call toLine
045B  E8 02F8 R                 call println
045E  FE C6                     inc dh
0460  74 02                     je page_up_date5
0462  E2 EB                     loop page_up_date3
0464                        page_up_date5:
0464  5A                       pop dx
0465  E8 0264 R                 call toLine
0468  2E: 8B 1E 0365 R          mov bx,cs:bx_save
046D  0A F6                     or dh,dh
046F  74 09                     jz page_up_date7
0471  8A CE                     mov cl,dh
0473  B5 00                     mov ch,0
0475                        page_up_date6:
0475  E8 05B7 R                 call NxtLine
0478  E2 FB                     loop page_up_date6
047A                        page_up_date7:
047A  07                       pop es
047B  1F                       pop ds
047C  5F                       pop di
047D  5E                       pop si
047E  5A                       pop dx
047F  59                       pop cx
0480  58                       pop ax
0481  C3                       ret
0482  C3                       ret
                            page_up_date endp
                            ;================== Link List Tools ============
                            =======
                            ; procedue dx use it to modify memory allocate b
                            efore
                            ; allocate other memory
```

```
                        ; and allocate memory for first line
0483                    init    proc near
0483  50                        push ax
0484  53                        push bx
0485  57                        push di
0486  06                        push es
0487  0E                        push cs
0488  07                        pop  es
0489  B4 4A                     mov ah,4ah
048B  BB 00CF                   mov bx,(offset seg_end - offset seg_begin +15 )
                         shr 4
048E  CD 21                     int 21h
0490  2E: 8C 0E 0167 R          mov cs:c1,cs
0495  E8 049D R                 call FirstLine
0498  07                        pop es
0499  5B                        pop bx
049A  5F                        pop di
049B  58                        pop ax
049C  C3                        ret
                        init    endp
049D                    FirstLine Proc near
049D  53                        push bx
049E  06                        push es
049F  57                        push di
04A0  E8 048F R                 call malloc
04A3  8C 06 0103 R              mov First,es
04A7  8C 06 0105 R              mov last,es
04AB  33 DB                     xor bx,bx
04AD  8B FB                     mov di,bx
04AF  26: 89 1D                 mov es:[di],bx
04B2  26: 89 5D 02              mov es:[di+2],bx
04B6  26: C6 45 04 00           mov byte ptr es:[di+4],0
04BB  5F                        pop di
04BC  07                        pop es
04BD  5B                        pop bx
04BE  C3                        ret
                        FirstLine Endp
                        ; malloc return new seg in es
                        ; error carry flag set
048F                    malloc proc near
04BF  50                        push ax
```

```
04C0  53                        push bx
04C1  B4 48                     mov ah,48h
04C3  BB 000B                   mov bx,(MaxLine+5+15)shr 4
04C6  CD 21                     int 21h
04C8  50                        push ax
04C9  07                        pop  es
04CA  5B                        pop  bx
04CB  58                        pop  ax
04CC  C3                        ret
                              malloc endp
                              ; input pinter in es
04CD                          release proc near
04CD  50                        push ax
04CE  06                        push es
04CF  B4 49                     mov ah,49h
04D1  CD 21                     int 21h
04D3  07                        pop  es
04D4  58                        pop  ax
04D5  C3                        ret
                              release endp
04D6                          ErrProc proc near
04D6  C3                        ret
                              ErrProc endp
                              ; get old line length
                              ; input bx pointer to the line
                              ; output in ch
04D7                          GetLeng proc near
04D7  57                        push di
04D8  06                        push es
04D9  8E C3                     mov es,bx
04DB  33 FF                     xor di,di
04DD  26: 8A 6D 04              mov ch,es:[di+4]
04E1  07                        pop  es
04E2  5F                        pop  di
04E3  C3                        ret
                              Getleng endp
                              ; Update line length
                              ; input bx pointer to the line ,ch update line l
                              ength
04E4                          CngLeng proc near
04E4  57                        push di
```

```
04E5  06                      push es
04E6  8E C3                   mov es,bx
04E8  33 FF                   xor di,di
04EA  26: 88 6D 04            mov es:[di+4],ch
04EE  07                      pop  es
04EF  5F                      pop  di
04F0  C3                      ret
                      CngLeng endp
                      ; use to allocate memory for a line and Insert i
                      t to the link lists
                      ; input pointer of line in bx
                      ; output bx will point to new line
04F1                  InsLine proc near
04F1  50                      push ax
04F2  56                      push si
04F3  57                      push di
04F4  06                      push es
04F5  1E                      push ds
04F6  E8 04BF R               call malloc
04F9  73 08                   jnc Ins_1
04FB  B0 01                   mov al,1
04FD  E8 04D6 R               call ErrProc
0500  EB 2B 90                jmp Ins_Ext
0503                  Ins_1:
0503  8E DB                   mov ds,bx ; ds point to update line
0505  33 FF                   xor di,di ; es point to New Line
0507  33 F6                   xor si,si ; clear pointer
0509  8B 44 02                mov ax,[si+2]
050C  26: 89 45 02            mov es:[di+2],ax
0510  8C 44 02                mov [si+2],es
0513  26: 89 1D               mov es:[di],bx
0516  2E: 3B 1E 0105 R        cmp bx,cs:last
051B  75 08                   jnz Ins_2
051D  2E; 8C 06 0105 R        mov cs:last,es
0522  EB 07 90                jmp Ins_3
0525                  Ins_2:
0525  26: 8E 5D 02            mov ds,es:[di+2]
0529  8C 04                   mov [si],es
052B                  Ins_3:
052B  8C C3                   mov bx,es
052D                  Ins_Ext:
```

```
052D  1F                        pop ds
052E  07                        pop es
052F  5F                        pop di
0530  5E                        pop si
0531  58                        pop ax
0532  C3                        ret
                        InsLine endp
                        ;input line to be erase in bx
                        ;output bx point to Nxt line
                        ;       ax point to Pre line
0533                    DelLine proc near
0533  56                        push si
0534  57                        push di
0535  1E                        push ds
0536  06                        push es
0537  8E C3                     mov es,bx
0539  33 FF                    xor di,di
053B  33 F6                     xor si,si
053D  26: 8B 05                 mov ax,es:[di]
0540  26: 8B 5D 02              mov bx,es:[di+2]
0544  E8 04CD R                 call release
0547  8E D8                     mov ds,ax
0549  89 5C 02                  mov [si+2],bx
054C  8E C3                     mov es,bx
054E  26: 89 05                 mov es:[di],ax
0551  07                        pop es
0552  1F                        pop ds
0553  5F                        pop di
0554  5E                        pop si
0555  C3                        ret
                        DelLine endp
0556                    DelAll proc near
0556  50                        push ax
0557  53                        push bx
0558  2E: 8B 1E 0103 R           mov bx,cs:first
055D                    DelAll2:
055D  EB 0533 R                 call DelLine
0560  0B DB                     or bx,bx
0562  74 02                     jz Delall1
0564  EB F7                     jmp delall2
0566                    DelAll1:
```

```
0566  2E: C7 06 0103 R 0000          mov cs:first,0
056D  2E: C7 06 0105 R 0000          mov cs:Last,0
0574  5B                            pop bx
0575  58                            pop ax
0576  C3                            ret
                                   Delall endp
0577                               InsBefore proc near
0577  50                                 push ax
0578  56                                 push si
0579  57                                 push di
057A  1E                                 push ds
057B  06                                 push es
057C  E8 04BF R                          call malloc
057F  73 08                              jnc InsB_1
0581  B0 00                              mov al,0
0583  E8 04D6 R                          call ErrProc
0586  EB 29 90                           Jmp InsBExit
0589  8E DB                         InsB_1: mov ds,bx
058B  33 F6                              xor si,si
058D  33 FF                              xor di,di
058F  8B 04                              mov ax,[si] ;
0591  26: 89 05                          mov es:[di],ax;New line point to Pre line
0594  26: 8C 5D 02                       mov es:[di+2],ds;New line point to Old line
0598  8C 04                              mov [si],es   ; old line point to New line
059A  2E: 3B 1E 0103 R                   cmp bx,cs:first ; Is it first line
059F  75 08                              jnz Insb_2
05A1  2E: 8C 06 0103 R                   mov cs:first,es
05A6  EB 07 90                           jmp Insb_3
05A9  26: 8E 1D                    Insb_2: mov ds,es:[di]
05AC  8C 44 02                           mov [si+2],es;pre line point to new line
05AF  8C C3                        Insb_3: mov bx,es
05B1                               InsbExit:
05B1  07                                 pop es
05B2  1F                                 pop ds
05B3  5F                                 pop di
05B4  5E                                 pop si
05B5  58                                 pop ax
05B6  C3                                 ret
                                   InsBefore endp
                                   ; point to Nxt line procedure
                                   ; input line pointer in bx,dx = row num
```

```
                        ; output  in bx , ZERO FLAG SET WHEN IT IS LAST
                        LINE
                        ;           dx = row num
05B7                    NxtLine proc near
05B7  06                        push es
05B8  57                        push di
05B9  8E C3                     mov es,bx
05BB  33 FF                     xor di,di
05BD  26: 8B 5D 02              mov bx,es:[di+2]
05C1  0B DB                     or bx,bx
05C3  75 02                     jnz NxtLine_1
05C5  8C C3                     mov bx,es
05C7                    NxtLine_1:
05C7  5F                        pop di
05C8  07                        pop  es
05C9  C3                        ret
                         NxtLine endp
05CA                     PreLine proc near
05CA  06                        push es
05CB  57                        push di
05CC  8E C3                     mov es,bx
05CE  33 FF                     xor di,di
05D0  26: 8B 1D                 mov bx,es:[di]
05D3  0B DB                     or bx,bx
05D5  75 02                     jnz PreLine_1
05D7  8C C3                     mov bx,es
05D9                    preLine_1:
05D9  5F                        pop di
05DA  07                        pop  es
05DB  C3                        ret
                        PreLine endp
                        ; input bx pointer
                        ; input cl order or char
                        ; output es:[di] point to it
                        ; not destroy any register except es and di
                        ; ZERO FLAG SET IF LINE LENG TO LONG
05DC                    char_line proc near
05DC  51                        push cx
05DD  80 F9 A0                  cmp cl,MaxLine
05E0  74 09                     jz char_exit
05E2  8E C3                     mov es,bx
```

```
05E4  BF 0005                   mov di,5
05E7  32 ED                     xor ch,ch
05E9  03 F9                     add di,cx
05EB                    char_exit:
05EB  59                        pop cx
05EC  C3                        ret
                        char_line endp
                        ; input bx,cl
                        ; output es:[di] point to char chang bx if in ne
                        xt line
05ED                    Next_char proc near
05ED  51                        push cx
05EE  E8 0407 R                 call getleng
05F1  3A E9                     cmp ch,cl
05F3  73 05                     jae Next_c1
05F5  E8 05B7 R                 call NxtLine
05F8  B1 00                     mov c1,0
05FA                    Next_c1:
05FA  E8 05DC R                 call char_line
05FD  59                        pop cx
05FE  C3                        ret
                        Next_char endp
                        ;================= Disk Control Area ==========
                        =============
05FF                    dir     proc near
05FF  50                        push ax
0600  51                        push cx
0601  52                        push dx
0602  B9 0000                   mov cx,0
0605  BA 0157 R                 mov dx,offset Comma
0608  B4 4E                     mov ah,4eh
060A  CD 21                     int 21h
060C  73 03                     jnc dir1
060E  E8 0615 R                 call PrintDir
0611                    dir1:
0611  5A                        pop dx
0612  59                        pop cx
0613  58                        pop ax
0614  C3                        ret
                        dir     endp
0615                    PrintDir proc near
```

```
0615  50                        push ax
0616  52                        push dx
0617  BA 0217 R                 mov dx,offset print_dir_1
061A  B4 09                     mov ah,9
061C  CD 21                     int 21h
061E  5A                        pop dx
061F  58                        pop ax
0620  C3                        ret
                          Printdir endp
0621                      File2asciiz proc near
0621  50                        push ax
0622  53                        push bx
0623  51                        push cx
0624  56                        push si
0625  BE 0193 R                 mov si,offset FileName
0628                      file2asc1:
062B  BA 07                     mov al,[bx]
062A  43                        inc bx
062B  3C 20                     cmp al,20h
062D  74 05                     jz file2asc2
062F  88 04                     mov [si],al
0631  46                        inc si
0632  EB F4                     jmp file2asc1
0634                      file2asc2:
0634  C6 04 2E                  mov byte ptr [si],'.'
0637  B1 03                     mov cl,3
0639                      file2asc3:
0639  8A 07                     mov al,[bx]
063B  88 04                     mov [si],al
063D  43                        inc bx
063E  46                        inc si
063F  FE C9                     dec cl
0641  75 F6                     jnz file2asc3
0643  5E                        pop si
0644  59                        pop cx
0645  5B                        pop bx
0646  58                        pop ax
0647  C3                        ret
                          file2asciiz endp
                          ; Input Cx num of byte to write
0648                      WriteRec proc near
```

```
0648  50                        push ax
0649  53                        push bx
064A  51                        push cx
064B  52                        push dx
064C  1E                        push ds
064D  0E                        push cs
064E  1F                        pop ds
064F  B4 40                     mov ah,40h
0651  8B 1E 018F R              mov bx,handle
0655  BA 06B6 R                 mov dx,offset line_Save
0658  CD 21                     int 21h
065A  1F                        pop ds
065B  5A                        pop dx
065C  59                        pop cx
065D  5B                        pop bx
065E  58                        pop ax
065F  C3                        ret
                        WriteRec endp
0660                    OpenFile proc near
0660  50                        push ax
0661  53                        push bx
0662  51                        push cx
0663  52                        push dx
0664  1E                        push ds
0665  0E                        push cs
0666  1F                        pop ds
0667  B4 3C                     mov ah,3ch
0669  BA 0193 R                 mov dx,offset fileName
066C  B9 0000                   mov cx,0
066F  CD 21                     int 21h
0671  2E: A3 018F R             mov CS:Handle,ax
0675  73 05                     Jnc OpenOk
0677  B0 01                     mov al,1
0679  E8 04D6 R                 call Errproc
067C                    OpenOK:
067C  1F                        pop ds
067D  5A                        pop dx
067E  59                        pop cx
067F  5B                        pop bx
0680  58                        pop ax
0681  C3                        ret
```

```
                        OpenFile endp
0682                    CloseFile proc near
0682  50                        push ax
0683  53                        push bx
0684  B4 3E                     mov ah,3eh
0686  2E: 8B 1E 018F R          mov bx,cs:handle
068B  CD 21                     int 21h
068D  5B                        pop bx
068E  58                        pop ax
068F  C3                        ret
                        CloseFile endp
0690                    SAve_help Proc Near
0690  56                        push si
0691  1E                        push ds
0692  0E                        push cs
0693  1F                        pop  ds
0694  B1 00                     mov cl,0
0696  E8 05DC R                 call Char_line
0699  BE 06B6 R                 mov si,offset line_save
069C  87 F7                     Xchg si,di
                        ;Exchange ds,es
069E  1E                        push ds
069F  06                        push es
06A0  1F                        pop ds
06A1  07                        pop es
06A2  E8 04D7 R                 call GetLeng
06A5  51                        push cx
06A6  8A CD                     mov cl,ch
06A8  B5 00                     mov ch,0
06AA  FC                        cld
06AB  F3/ A4                    rep movsb
06AD  59                        pop cx
06AE  8A CD                     mov cl,ch
06B0  B5 00                     mov ch,0
06B2  41                        inc cx
06B3  1F                        pop ds
06B4  5E                        pop si
06B5  C3                        ret
                        Save_help endp
06B6  0200[             Line_Save db 512 dup(?)
        ??
```

```
                    ]

08B6                        SaveToDisk proc near
08B6  50                            push ax
08B7  53                            push bx
08B8  51                            push cx
08B9  52                            push dx
08BA  57                            push di
08BB  56                            push si
08BC  1E                            push ds
08BD  06                            push es
08BE  0E                            push cs
08BF  1F                            pop ds
08C0  E8 0660 R                     call openfile
08C3  72 26                         Jc SaveToDiskEXit
08C5  2E: 8B 1E 0103 R              mov bx,cs:first
08CA                        Save_1:
08CA  E8 0690 R                     call Save_help
08CD  26: C6 05 0A                  mov es:[di],byte ptr 0ah ;lf
08D1  E8 0648 R                     call WriteRec
08D4  72 15                         jc SaveTODiskExit
08D6  E8 05B7 R                     call NxtLine
08D9  74 02                         jz Save_2
08DB  EB ED                         jmp Save_1
08DD                        SAve_2:
08DD  C6 06 06B6 R 1A               mov line_save,01ah
08E2  B9 0001                       mov cx,1
08E5  E8 0648 R                     call WriteRec
08E8  E8 0682 R                     call CloseFile
08EB                        SaveToDiskExit:
08EB  07                            pop es
08EC  1F                            pop ds
08ED  5E                            pop si
08EE  5F                            pop di
08EF  5A                            pop dx
08F0  59                            pop cx
08F1  5B                            pop bx
08F2  58                            pop ax
08F3  C3                            ret
                            SaveToDisk endp
08F4                        OpenRd proc near
```

```
08F4  50                        push ax
08F5  52                        push dx
08F6  1E                        push ds
08F7  B4 3D                     mov ah,3dh
08F9  8D 16 0193 R              lea dx,FileName
08FD  CD 21                     int 21h
08FF  A3 018F R                 mov handle,ax
0902  1F                        pop ds
0903  5A                        pop dx
0904  58                        pop ax
0905  C3                        ret
                        OpenRd endp
0906  0000               datacnt dw 0
0908                    Readf proc near
0908  50                        push ax
0909  53                        push bx
090A  51                        push cx
090B  52                        push dx
090C  1E                        push ds
090D  0E                        push cs
090E  1F                        pop  ds
090F  B4 3F                     mov ah,3fh
0911  8B 1E 018F R              mov bx,handle
0915  BA 06B6 R                 mov dx,offset line_save
0918  B9 0200                   mov cx,512
091B  CD 21                     int 21h
091D  A3 0906 R                 mov datacnt ,ax
0920  1F                        pop ds
0921  5A                        pop dx
0922  59                        pop cx
0923  5B                        pop bx
0924  58                        pop ax
0925  C3                        ret
                        Readf endp
                        ;output in al
0926                    Load proc near
0926  50                        push ax
0927  53                        push bx
0928  51                        push cx
0929  56                        push si
092A  57                        push di
```

```
092B  1E                          push ds
092C  06                          push es
092D  0E                          push cs
092E  1F                          pop  ds
092F  E8 08F4 R                   call openrd
0932  73 08                       jnc load1
0934  B0 02                       mov al,2
0936  E8 04D6 R                   call ErrProc
0939  EB 6E 90                    jmp LoadExt
093C                        load1:
093C  E8 0556 R                   call Delall
093F  E8 049D R                   call firstLine
0942  8B 1E 0103 R                mov bx,first
0946  B1 00                       mov cl,0
0948  E8 05DC R                   call char_line
094B  C7 06 0906 R 0000           mov datacnt,0
0951  B5 00                       mov ch,0
0953  BE 06B6 R                   mov si,offset line_save
0956  E8 0908 R                   call readf
0959  73 03                       jnc loadloop
095B  EB 46 90                    jmp load5
095E                        loadloop:
095E  8A 04                       mov al,[si]
0960  3C 1A                       cmp al,1ah
0962  74 3F                       jz load5
0964  46                          inc si
0965  FF 0E 0906 R                dec datacnt
0969  75 0B                       jnz load2
096B  BE 06B6 R                   mov si,offset line_save
096E  E8 0908 R                   call readf
0971  73 03                       jnc load2
0973  EB 2E 90                    jmp load5
0976                        load2:
0976  26: 88 05                   mov es:[di],al
0979  47                          inc di
097A  FE C5                       inc ch
097C  3C 0D                       cmp al,13
097E  74 02                       jz load3
0980  EB DC                       jmp loadloop
0982  46                    load3:  inc si
0983  FF 0E 0906 R                dec datacnt
```

```
09B7  75 0B                     jnz load4
09B9  BE 06B6 R                 mov si,offset line_save
09BC  E8 0908 R                 call readf
09BF  73 03                     jnc load4
09C1  EB 10 90                  jmp load5
09C4                        load4:
09C4  E8 04E4 R                 call cngleng
09C7  E8 04F1 R                 call insLine
09CA  B1 00                     mov cl,0
09CC  E8 05DC R                 call char_line
09CF  B5 00                     mov ch,0
09D1  EB BB                     jmp loadloop
09D3                        load5:
09D3  E8 0533 R                 call DelLine
09D6  E8 0682 R                 call closeFile
09D9                        LoadExt:
09D9  07                        pop es
09DA  1F                        pop ds
09DB  5F                        pop di
09DC  5E                        pop si
09DD  59                        pop cx
09DE  5B                        pop bx
09DF  58                        pop ax
09E0  C3                        ret
                            Load endp
                            ;====================== Print Utility ==========
                            =============
                            ;Print a decimal number
                            ; input in dx
09E1                        DecPrint proc near
09E1  50                        push ax
09E2  51                        push cx
09E3  52                        push dx
09E4  56                        push si
09E5  8B C2                     mov ax,dx
09E7  BE 000A                   mov si,10
09EA  33 C9                     xor cx,cx
09EC                        decprint1:
09EC  33 D2                     xor dx,dx
09EE  F7 F6                     div si
09F0  52                        push dx
```

```
09C1  41                        inc cx
09C2  0B C0                     or ax,ax
09C4  75 F6                     jne decprint1
09C6                    decprint2:
09C6  5A                        pop dx
09C7  80 C2 30                  add dl,30h
09CA  B4 02                     mov ah,2
09CC  CD 21                     int 21h
09CE  E2 F6                     loop decprint2
09D0  5E                        pop si
09D1  5A                        pop dx
09D2  59                        pop cx
09D3  58                        pop ax
09D4  C3                        ret
                        DecPrint endp
                        Hexprint Proc near
09D5  50                        push ax
09D6  51                        push cx
09D7  B4 02                     mov ah,2
09D9  B5 04                     mov ch,4
09DB                    Hex1:
09DB  B1 04                     mov cl,4
09DD  D3 C2                     rol dx,cl
09DF  52                        push dx
09E0  80 E2 0F                  and dl,0fh
09E3  80 C2 30                  add dl,30h
09E6  80 FA 3A                  cmp dl,3ah
09E9  7C 03                     jl Hex2
09EB  80 C2 07                  add dl,7h
09EE                    Hex2:
09EE  CD 21                     int 21h
09F0  5A                        pop dx
09F1  FE CD                     dec ch
09F3  75 E6                     jnz hex1
09F5  59                        pop cx
09F6  58                        pop ax
09F7  C3                        ret
                        HexPrint endp
                        ;==================== Shell Command ============
                        ============
09F8                    Dos_Com proc near
```

```
09F8  51                        push cx
09F9  56                        push si
09FA  57                        push di
09FB  1E                        push ds
09FC  06                        push es
09FD  0E                        push cs
09FE  0E                        push cs
09FF  1F                        pop ds
0A00  07                        pop es
0A01  C6 06 011E R 00           mov cmd_tail,0
0A06  E8 0AA4 R                 call shell
0A09  07                        pop es
0A0A  1F                        pop ds
0A0B  5F                        pop di
0A0C  5E                        pop si
0A0D  59                        pop cx
0A0E  C3                        ret
                        Dos_com endp
0A0F                    get_cmd_line proc near
0A0F  50                        push ax
0A10  53                        push bx
0A11  52                        push dx
0A12  1E                        push ds
0A13  0E                        push cs
0A14  1F                        pop  ds
0A15  BA 010C R                 mov dx,offset dos_prompt
0A18  B4 09                     mov ah,9
0A1A  CD 21                     int 21h
0A1C  C6 06 0120 R 1E           mov com_buff,30
0A21  B4 0A                     mov ah,0ah
0A23  BA 0120 R                 mov dx,offset com_buff
0A26  CD 21                     int 21h
0A28  BA 1E 0121 R              mov bl,com_buff+1
0A2C  B7 00                     mov bh,0
0A2E  88 1E 011E R              mov cmd_tail,bl
0A32  0A DB                     or bl,bl
0A34  74 12                     jz get_cmd_1
0A36  C6 87 0122 R 0D           mov [com_buff+bx+2],0dh;insert carriage return
0A3B  C6 06 0121 R 20           mov com_buff+1,20h
0A40  C6 06 0120 R 63           mov com_buff,'c'
0A45  EB 15 90                  jmp get_cmd_2
```

```
0A4B                            get_cmd_1:
0A4B  EB 0241 R                         call cls
0A4B  52                                push dx
0A4C  BA 0100                           mov dx,100h
0A4F  E8 0259 R                         call gotoxy
0A52  50                                push ax
0A53  B4 09                             mov ah,9
0A55  BA 0A61 R                         mov dx,offset get_cmd_mess
0A58  CD 21                             int 21h
0A5A  58                                pop ax
0A5B  5A                                pop dx
0A5C                            get_cmd_2:
0A5C  1F                                pop ds
0A5D  5A                                pop dx
0A5E  5B                                pop bx
0A5F  58                                pop ax
0A60  C3                                ret
0A61  54 4F 20 44 4F 53 20      get_cmd_mess db "TO DOS SHELL TYPE 'EXIT' WITH P
      53 48 45 4C 4C 20 54      RESS [RETURN] BACK TO THE PROGRAM",0DH,0AH,'$'
      59 50 45 20 27 45 58
      49 54 27 20 57 49 54
      48 20 50 52 45 53 53
      20 5B 52 45 54 55 52
      4E 5D 20 42 41 43 4B
      20 54 4F 20 54 48 45
      20 50 52 4F 47 52 41
      4D 0D 0A 24

                                get_cmd_line endp
0AA4                            shell proc near
0AA4  50                                push ax
0AA5  53                                push bx
0AA6  51                                push cx
0AA7  52                                push dx
0AA8  56                                push si
0AA9  57                                push di
0AAA  1E                                push ds
0AAB  06                                push es
0AAC  55                                push bp
0AAD  0E                                push cs
0AAE  0E                                push cs
```

```
0AAF  07                        pop es
0AB0  1F                        pop ds
0AB1  89 26 0153 R              mov stackptr,sp
0AB5  8C 16 0155 R              mov stackseg,ss
0AB9  B4 4B                     mov ah,4bh
0ABB  B0 00                     mov al,0
0ABD  BA 0157 R                 mov dx,offset comma
0AC0  BB 0163 R                 mov bx,offset parameter
0AC3  CD 21                     int 21h
0AC5  2E: 8B 26 0153 R          mov sp ,cs:stackptr
0ACA  2E: 8E 16 0155 R          mov ss ,cs:stackseg
0ACF  5D                        pop  bp
0AD0  07                        pop  es
0AD1  1F                        pop  ds
0AD2  5F                        pop  di
0AD3  5E                        pop  si
0AD4  5A                        pop  dx
0AD5  59                        pop  cx
0AD6  5B                        pop  bx
0AD7  58                        pop  ax
0AD8  C3                        ret
                        shell endp
0AD9  50 72 65 73 73 20 41      to_shell_mess db 'Press Any Press Back To The Pr
                                ogram $'
      6E 79 20 50 72 65 73
      73 20 42 61 63 6B 20
      54 6F 20 54 68 65 20
      50 72 6F 67 72 61 6D
      20 24
0AFE                            to_shell proc near
0AFE  50                              push ax
0AFF  53                              push bx
0B00  52                              push dx
0B01  E8 0275 R                       call GetLine
0B04  52                              push dx
0B05  53                              push bx
0B06  E8 0241 R                       call cls
0B09  BA 1800                         mov dx,1800h
0B0C  E8 0259 R                       call GotoXy
0B0F  E8 0A0F R                       call get_cmd_line
0B12  E8 0AA4 R                       call shell
```

```
0B15  50                              push ax
0B16  52                              push dx
0B17  BA 0000                         mov dx,0
0B1A  E8 0259 R                       call gotoxy
0B1D  BA 0AD9 R                       mov dx,offset to_shell_mess
0B20  B4 09                           mov ah,9
0B22  CD 21                           int 21h
0B24  E8 0B88 R                       call pause_a
0B27  5A                              pop dx
0B28  58                              pop  ax
0B29  5B                              pop bx
0B2A  5A                              pop dx
0B2B  E8 0264 R                       call ToLine
0B2E  E8 0427 R                       call page_up_date
0B31  5A                              pop  dx
0B32  5B                              pop  bx
0B33  58                              pop ax
0B34  C3                              ret
                              to_shell endp
= 000E                        dsk equ 0eh
0B35                          set_shell proc near
0B35  50                              push ax
0B36  8E 06 002C R                    mov es,env_seg
0B3A  26: C6 06 000E 63               mov byte ptr es:[dsk],'c'
0B40  58                              pop ax
0B41  C3                              ret
                              set_shell endp
0B42  4D 45 4E 55 20 4C 49    data_test db 'MENU LINE IN 0 ROW ',0
      4E 45 20 49 4E 20 30
      20 52 4F 57 20 00
0B56  73 61 76 65 20 74 65    data_test_2 db 'save test 2nd line',13,0ffh
      73 74 20 32 6E 64 20
      6C 69 6E 65 0D FF
0B6A  73 61 76 65 20 74 65    data_test_3 db 'save test 3rd line',13,0ffh
      73 74 20 33 72 64 20
      6C 69 6E 65 0D FF
0B7E  49                      key_table db 73
0B7F  0367 R                            dw offset page_up
0B81  51                                db 81
0B82  03BF R                            dw offset page_down
0B84  44                                db 68
```

```
0B85  0AFE R                    dw to_shell
0B87  00                        db 00
0B88                    pause_a proc near
0B88  B4 08                     mov ah,8
0B8A  CD 21                     int 21h
0B8C  C3                        ret
                        pause_a endp
0B8D                    skiphead:
                        ;====================== Begin Main =============
                        ===============
0B8D  BC 0CE3 R                 mov sp,offset seg_end
0B90  E8 0483 R                 call init
0B93  E8 0B35 R                 call set_shell
0B96  E8 02A0 R                 call set_screen
0B99  E8 0241 R                 call cls
0B9C  E8 0926 R                 call load
0B9F  2E: 8B 1E 0103 R          mov bx,cs:first
0BA4  B9 000A                   mov cx,10
0BA7                    sk:
0BA7  E8 05B7 R                 call Nxtline
0BAA  E2 FB                     loop sk
0BAC  BA 0000                   mov dx,0
0BAF  E8 0264 R                 call toLine
0BB2  B9 0008                   mov cx,8
0BB5                    sk2:
0BB5  0A F6                     or dh,dh
0BB7  74 06                     jz sk3
0BB9  E8 0264 R                 call toLine
0BBC  E8 05B7 R                 call Nxtline
0BBF                    sk3:
0BBF  E8 0C30 R                 call print
0BC2  FE C6                     inc dh
0BC4  B2 00                     mov dl,0
0BC6  E2 ED                     loop sk2
0BC8  FE CE                     dec dh
0BCA  E8 05CA R                 call preline
0BCD  FE CE                     dec dh
0BCF  E8 05CA R                 call preline
0BD2  FE CE                     dec dh
0BD4  E8 05CA R                 call preline
0BD7  FE CE                     dec dh
```

```
0BD9  E8 05CA R                 call preline
0BDC  FE CE                     dec dh
0BDE  E8 05CA R                 call preline
0BE1  FE CE                     dec dh
0BE3  E8 05CA R                 call preline
0BE6  FE CE                     dec dh
0BE8  B2 32                     mov dl,50
0BEA  E8 0264 R                 call toline
0BED                        sk4:
0BED  E8 0B88 R                 call pause_a
0BF0  0A C0                     or al,al
0BF2  74 06                     jz skf
0BF4  3C 0D                     cmp al,13
0BF6  74 1C                     jz skexit
0BF8  EB F3                     jmp sk4
0BFA  E8 0B88 R             skf:    call pause_a
0BFD  BE 0B7E R                 mov si,offset key_table
0C00                        skf1:
0C00  80 3C 00                  cmp byte ptr [si],0
0C03  74 E8                     jz sk4
0C05  3A 04                     cmp al,[si]
0C07  74 05                     jz skff
0C09  83 C6 03                  add si,3
0C0C  EB F2                     jmp skf1
0C0E                        skff:
0C0E  46                        inc si
0C0F  2E: FF 14                 call cs:[si]
0C12  EB D9                     jmp sk4
0C14                        skexit:
0C14  BA 04D2                   mov dx,1234
0C17  E8 09B1 R                 call Decprint
0C1A  B4 4C                     mov ah,4ch
0C1C  CD 21                     int 21h

0C1E  74 6F 20 73 68 65 6C  mmm db 'to shell command $'
      6C 20 63 6F 6D 6D 61
      6E 64 20 24

                            ;==================== Test Utility ============
                            =========
0C30                        print proc near
0C30  51                        push cx
```

```
0C31  50                        push ax
0C32  52                        push dx
0C33  57                        push di
0C34  06                        push es
0C35  B1 00                     mov cl,0
0C37  E8 05DC R                 call char_line
0C3A  E8 04D7 R                 call getleng
0C3D  B4 02                     mov ah,2
0C3F                    print1:
0C3F  26: 8A 15                 mov dl,es:[di]
0C42  47                        inc di
0C43  CD 21                     int 21h
0C45  FE C9                     dec ch
0C47  75 F6                     jnz print1
0C49  B2 0D                     mov dl,13
0C4B  CD 21                     int 21h
0C4D  B2 0A                     mov dl,0ah
0C4F  CD 21                     int 21h
0C51  07                        pop es
0C52  5F                        pop di
0C53  5A                        pop dx
0C54  58                        pop ax
0C55  59                        pop cx
0C56  C3                        ret
                        print   endp
0C57                    put     proc near
0C57  51                        push cx
0C58  B5 00                     mov ch,0
0C5A  B1 00                     mov cl,0
0C5C  E8 05DC R                 call char_line
0C5F                    next:
0C5F  8A 04                     mov al,[si]
0C61  26: 88 05                 mov es:[di],al
0C64  3C FF                     cmp al,0ffh
0C66  74 06                     jz kk
0C68  46                        inc si
0C69  47                        inc di
0C6A  FE C5                     inc ch
0C6C  EB F1                     jmp next
0C6E  E8 04E4 R         kk:     call cngleng
0C71  59                        pop cx
```

```
0C72  C3                        ret
                        put     endp
0C73  0010[                     db 16 dup('stack  ')
          73 74 61 63 6B
             20 20
                      ]

0CE3                            seg_end:
0CE3                            code ends
                                end start
```

Segments and Groups:

Name	Size	Align	Combine Class
CODE	0CE3	PARA	NONE

Symbols:

Name	Type	Value	Attr	
BX_SAVE	L WORD	0365	CODE	
C1	L WORD	0167	CODE	
CHAR_EXIT	L NEAR	05EB	CODE	
CHAR_LINE	N PROC	05DC	CODE	Length = 0011
CLOSEFILE	N PROC	0682	CODE	Length = 000E
CLS	N PROC	0241	CODE	Length = 0018
CMD_TAIL	L BYTE	011E	CODE	
CNGLENG	N PROC	04E4	CODE	Length = 000D
COL	L WORD	0109	CODE	
COMMA	L BYTE	0157	CODE	
COM_BUFF	L BYTE	0120	CODE	
DATACNT	L WORD	0906	CODE	
DATA_TEST	L BYTE	0842	CODE	
DATA_TEST_2	L BYTE	0B56	CODE	
DATA_TEST_3	L BYTE	0B6A	CODE	
DECPRINT	N PROC	09B1	CODE	Length = 0024
DECPRINT1	L NEAR	09BC	CODE	
DECPRINT2	L NEAR	09C6	CODE	
DELALL	N PROC	0556	CODE	Length = 0021
DELALL1	L NEAR	0566	CODE	
DELALL2	L NEAR	055D	CODE	
DELLINE	N PROC	0533	CODE	Length = 0023
DIR	N PROC	05FF	CODE	Length = 0016
DIR1	L NEAR	0611	CODE	
DOS_COM	N PROC	09F8	CODE	Length = 0017
DOS_PROMPT	L BYTE	010C	CODE	
DSK	Number	000E		
DTA	L BYTE	0080	CODE	

ENV_SEG	L WORD	002C	CODE	
ERRPROC	N PROC	04D6	CODE	Length = 0001
FILE2ASC1	L NEAR	062B	CODE	
FILE2ASC2	L NEAR	0634	CODE	
FILE2ASC3	L NEAR	0639	CODE	
FILE2ASCIIZ	N PROC	0621	CODE	Length = 0027
FILENAME	L BYTE	0193	CODE	
FILE_1	L BYTE	01AF	CODE	Length = 000D
FILE_2	L BYTE	01BC	CODE	Length = 000D
FILE_3	L BYTE	01C9	CODE	Length = 000D
FILE_4	L BYTE	01D6	CODE	Length = 000D
FILE_5	L BYTE	01E3	CODE	Length = 000D
FILE_6	L BYTE	01F0	CODE	Length = 000D
FILE_7	L BYTE	01FD	CODE	Length = 000D
FILE_8	L BYTE	020A	CODE	Length = 000D
FILE_FOUND	L BYTE	009E	CODE	
FILE_NAME	L BYTE	0191	CODE	
FIRST	L WORD	0103	CODE	
FIRSTLINE	N PROC	049D	CODE	Length = 0022
GETLENG	N PROC	04D7	CODE	Length = 000D
GETLINE	N PROC	0275	CODE	Length = 002B
GETXY	N PROC	0234	CODE	Length = 000D
GET_CMD_1	L NEAR	0A48	CODE	
GET_CMD_2	L NEAR	0A5C	CODE	
GET_CMD_LINE	N PROC	0A0F	CODE	Length = 0095
GET_CMD_MESS	L BYTE	0A61	CODE	
GET_LI	L NEAR	0289	CODE	
GET_LINE1	L NEAR	0293	CODE	
GET_LINE2	L NEAR	0296	CODE	
GET_LINE_MESS	L BYTE	0174	CODE	
GOTOXY	N PROC	0259	CODE	Length = 000B
HANDLE	L WORD	018F	CODE	
HEX1	L NEAR	09DB	CODE	
HEX2	L NEAR	09EE	CODE	
HEXPRINT	N PROC	09D5	CODE	Length = 0023
INIT	N PROC	0483	CODE	Length = 001A
INSBEFORE	N PROC	0577	CODE	Length = 0040

INSBEXIT	L NEAR	05B1	CODE	
INSB_1	L NEAR	0589	CODE	
INSB_2	L NEAR	05A9	CODE	
INSB_3	L NEAR	05AF	CODE	
INSLINE	N PROC	04F1	CODE	Length = 0042
INS_1	L NEAR	0503	CODE	
INS_2	L NEAR	0525	CODE	
INS_3	L NEAR	052B	CODE	
INS_EXT	L NEAR	052D	CODE	
KEY_TABLE	L BYTE	0B7E	CODE	
KK	L NEAR	0C6E	CODE	
LAST	L WORD	0105	CODE	
LINE_SAVE	L BYTE	06B6	CODE	Length = 0200
LOAD	N PROC	0926	CODE	Length = 008B
LOAD1	L NEAR	093C	CODE	
LOAD2	L NEAR	0976	CODE	
LOAD3	L NEAR	0982	CODE	
LOAD4	L NEAR	0994	CODE	
LOAD5	L NEAR	09A3	CODE	
LOADEXT	L NEAR	09A9	CODE	
LOADLOOP	L NEAR	095E	CODE	
MALLOC	N PROC	04BF	CODE	Length = 000E
MAXLINE	Number	00A0		
MMM	L BYTE	0C1E	CODE	
MODE_SC	L BYTE	0173	CODE	
NEXT	L NEAR	0C5F	CODE	
NEXT_C1	L NEAR	05FA	CODE	
NEXT_CHAR	N PROC	05ED	CODE	Length = 0012
NXTLINE	N PROC	05B7	CODE	Length = 0013
NXTLINE_1	L NEAR	05C7	CODE	
OPENFILE	N PROC	0660	CODE	Length = 0022
OPENOK	L NEAR	067C	CODE	
OPENRD	N PROC	08F4	CODE	Length = 0012
PAGE_DOWN	N PROC	03BF	CODE	Length = 006B
PAGE_DOWN1	L NEAR	03D2	CODE	

PAGE_DOWN2	L NEAR	03D9	CODE	
PAGE_DOWN3	L NEAR	03E4	CODE	
PAGE_DOWN4	L NEAR	03EF	CODE	
PAGE_DOWN5	L NEAR	03FB	CODE	
PAGE_DOWN6	L NEAR	040C	CODE	
PAGE_DOWN7	L NEAR	0411	CODE	
PAGE_DOWN8	L NEAR	0419	CODE	
PAGE_UP	N PROC	0367	CODE	Length = 0058
PAGE_UP1	L NEAR	037A	CODE	
PAGE_UP2	L NEAR	0381	CODE	
PAGE_UP3	L NEAR	038C	CODE	
PAGE_UP4	L NEAR	0395	CODE	
PAGE_UP5	L NEAR	03A1	CODE	
PAGE_UP6	L NEAR	03B2	CODE	
PAGE_UP7	L NEAR	03B7	CODE	
PAGE_UP_DATE	N PROC	0427	CODE	Length = 005C
PAGE_UP_DATE1	L NEAR	043D	CODE	
PAGE_UP_DATE2	L NEAR	0444	CODE	
PAGE_UP_DATE3	L NEAR	044F	CODE	
PAGE_UP_DATE4	L NEAR	045B	CODE	
PAGE_UP_DATE5	L NEAR	0464	CODE	
PAGE_UP_DATE6	L NEAR	0475	CODE	
PAGE_UP_DATE7	L NEAR	047A	CODE	
PARAMETER	L WORD	0163	CODE	
PAUSE_A	N PROC	0B88	CODE	Length = 0005
PRELINE	N PROC	05CA	CODE	Length = 0012
PRELINE_1	L NEAR	05D9	CODE	
PRINT	N PROC	0C30	CODE	Length = 0027
PRINT1	L NEAR	0C3F	CODE	
PRINTDIR	N PROC	0615	CODE	Length = 000C
PRINTLN	N PROC	02F8	CODE	Length = 006D
PRINT_DIR_1	L BYTE	0217	CODE	
PRLN0	L NEAR	0335	CODE	
PRLN00	L NEAR	0322	CODE	
PRLN1	L NEAR	034A	CODE	
PRLN2	L NEAR	035C	CODE	
PUT	N PROC	0C57	CODE	Length = 001C
READF	N PROC	0908	CODE	Length = 001E
RELEASE	N PROC	04CD	CODE	Length = 0009
ROW	L WORD	0107	CODE	

SAVETODISK	N PROC	08B6	CODE	Length = 003E
SAVETODISKEXIT	L NEAR	08EB	CODE	
SAVE_1	L NEAR	08CA	CODE	
SAVE_2	L NEAR	08DD	CODE	
SAVE_HELP	N PROC	0690	CODE	Length = 0026
SCREEN	L WORD	0171	CODE	
SEG_BEGIN	L NEAR	0000	CODE	
SEG_END	L NEAR	0CE3	CODE	
SET_SC1	L NEAR	02BE	CODE	
SET_SCREEN	N PROC	02A0	CODE	Length = 0022
SET_SHELL	N PROC	0B35	CODE	Length = 000D
SHELL	N PROC	0AA4	CODE	Length = 0035
SK	L NEAR	0BA7	CODE	
SK2	L NEAR	0BB5	CODE	
SK3	L NEAR	0BBF	CODE	
SK4	L NEAR	0BED	CODE	
SKEXIT	L NEAR	0C14	CODE	
SKF	L NEAR	0BFA	CODE	
SKF1	L NEAR	0C00	CODE	
SKFF	L NEAR	0C0E	CODE	
SKIPHEAD	L NEAR	0B8D	CODE	
STACKPTR	L WORD	0153	CODE	
STACKSEG	L WORD	0155	CODE	
START	L NEAR	0100	CODE	
TOLINE	N PROC	0264	CODE	Length = 0011
TO_SHELL	N PROC	0AFE	CODE	Length = 0037
TO_SHELL_MESS	L BYTE	0AD9	CODE	
WRITEREC	N PROC	0648	CODE	Length = 0018
WRITE_DI1	L NEAR	02EE	CODE	
WRITE_DIRECT	N PROC	02C2	CODE	Length = 0036
WRITE_WAIT1	L NEAR	02E4	CODE	
WRITE_WAIT2	L NEAR	02E9	CODE	
X	L BYTE	0108	CODE	

1311 Source Lines
1311 Total Lines
195 Symbols

43598 Bytes symbol space free

0 Warning Errors
0 Severe Errors

กิตติกรรมประกาศ

ปริญญานิพนธ์ ฉบับนี้สำเร็จตามวัตถุประสงค์ได้ ก็โดยความช่วยเหลือและคำแนะนำ
อย่างดีจาก อาจารย์ รัตติกร วรากูลศิริพันธุ์ จึงขอขอบพระคุณ ณ.ที่นี้ด้วย

ก. เอกสารอ้างอิงเป็นภาษาไทย

1. กลุ่ม เจเอสเค " แอสแซมบลี้ 8086/8088 ," สำนักพิมพ์พิสิกส์เซนเตอร์ พศ 2530

2. อาจารย์ ยืน ภู่วรวรรณ ,"โปรแกรมคอมพิวเตอร์ ภาษแอสเซมบลี้ 8086/8088 " ซีเอ็ดยูเคชั่น พศ 2529

ข. เอกสารอ้างอิงที่เป็นภาษาอังกฤษ

1. David C. Willen and Jeffrey I. Krantz ,"8088 assemblen language programing: the IBM PC " Howard W. Sam & Co 1983

2. Ray Duncan "Advanced MS DOS" osoft Press A Division of Microsoft Corporat 1986