ปีการศึกษา 2532

คอมพิวเตอร์กราฟฟิคส์
(COMPUTER GRAPHICS)

โดย

นายยงยุทธ เตียนโพธิ์ทอง 291167
นายวิบูลย์ ประมวลอุรุรัตน์ 291174

อาจารย์ที่ปรึกษา

รศ. มนัส สังวรศิลป์

ปริญญานิพนธ์ปีการศึกษา 2532

ภาควิชาอิเล็กโทรนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

เรื่อง คอมพิวเตอร์กราฟฟิคส์

ผู้จัดทำ

1 นายยงยุทธ เตียนโพธิ์ทอง 291167

2 นายวิบูลย์ ประมวลอุรุรัตน์ 291174

.....อาจารย์ที่ปรึกษา

(.....)

คอมพิวเตอร์กราฟิค

(COMPUTER GRAPHIC)

นายยงยุทธ เตียนโพธิ์ทอง 29.1167
นายวิบูลย์ ประมวลอุรุรัตน์ 29.1174
อาจารย์ที่ปรึกษา:
รศ. มนัส สังวรศิลป์
ปีการศึกษา 2532

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้มีเนื้อหาเกี่ยวกับ การพัฒนาระบบแฟ้มข้อมูลภาพบนไมโครคอมพิวเตอร์ ซึ่งเก็บภาพ(IMAGE FILE) และข้อความที่เป็นรายละเอียดของภาพ(TEXT) แยกกัน โดยจะแสดงผลบนจอมอนิเตอร์(MONITOR) ที่มีความละเอียด 512*256 จุดภาพ (PIXEL) ที่ระดับความเข้ม 256 ระดับเท่า

นอกจากนี้ยังจะกล่าวถึงความรู้พื้นฐานต่างๆ ที่ใช้ในการทำแฟ้มข้อมูล อาทิเช่น ระบบการแสดงผล ซึ่งประกอบด้วยอุปกรณ์การแสดงผลการประมวลผลแบบตัวอักษร และการประมวลผลแบบกราฟิค การโหลด (LOAD) ภาพ การเซฟ (SAVE) ภาพ การโหลดฟอนต์ (FONT) การเขียนตัวอักษรลงบนภาพในกราฟิคโหมด (Graphics Mode)

ถึงแม้ว่าการเก็บแฟ้มข้อมูลด้วยภาพนี้จะทำให้มีความสะดวกในการใช้งานและสื่อสารได้ดีกว่าแฟ้มข้อมูลแบบตัวอักษรก็ตาม แต่ก็มีปัญหาในเรื่องที่แต่ละภาพต้องใช้เนื้อที่ในการเก็บมาก ซึ่งแก้ไขโดยการใช้วิธี "การหาขอบภาพ" (EDGE DETECTION) หาขอบภาพแล้วตัดแบ็คกราวด์ (BACKGROUND) ออก ด้วยวิธีนี้จะสามารถประหยัดเนื้อที่ในการเก็บข้อมูลไปได้มากที่เดียว

โครงงานนี้สามารถนำไปประยุกต์ใช้กับ การเก็บข้อมูลขององค์ประกอบบนใบหน้าเพื่อใช้ในการสเก็ตภาพคนร้ายหรือการปรับปรุงภาพถ่ายจากดาวเทียม

COMPUTER GRAPHICS

YONGYUT TIANPOTONG 291167

WIBOON PRAMUAN-URURAT 291174

ASSOCIATE PROFESSOR MANUS SUNGWORASIL ADVISOR

1989

Abstract

This thesis has details about the development of image-database file system on micro-computer which save image file and database file separately. Data from image file and database file will display simultaneously on monitor which has resolution of 512 by 256 pixels , 256 grey levels.

Besides it will talk about the basic knowledge in dealing with image-database files. ie. display system , alphanumeric processing , graphics processing , loading image , saving image , loading font and writing characters on image in Graphics Modes.

Although this image-database file system is more convenient in using and communication than only database files (don't have image files) , but there is a problem about wasting bulky area in saving image files. This problem is alleviated by using " EDGE-DETECTION " , detect edge of image then cut background away. By this method it will economize area in saving image files.

This project can bring to utilize in other application. ie. , saving data of composition on face (ie. , eyes , mouth , nose , etc.) which use in sketch criminal face , the improvement of photograph from satellite.

สารบัญ

บทที่ 1 บทนำ.....1

บทที่ 2 ความรู้ทั่วๆไปเกี่ยวกับอีจีเอ/วีจีเอ.....4

2.1 ลักษณะทั่วๆไปของอีจีเอ/วีจีเอ.....4

2.2 หน่วยความจำแสดงผล.....4

2.3 โหมดตัวอักษร.....4

2.4 โหมดกราฟฟิคส์.....5

2.5 รีจิสเตอร์ควบคุม.....5

2.6 โหมดการแสดงผล.....6

2.7 ความละเอียดของภาพ.....6

2.8 ความละเอียดของสี.....6

2.9 พาเล็ทสีและรีจิสเตอร์สี.....8

2.10 การประมวลผลทางตัวอักษร.....10

2.11 การประมวลผลทางกราฟฟิคส์.....14

บทที่ 3 การหาขอบภาพ.....22

3.1 ตัวดำเนินการเกรเดียนท์.....23

3.2 ตัวดำเนินการคอมพาส.....24

3.3 ตัวดำเนินการลาปาซและการผ่านจุดศูนย์.....26

บทที่ 4 การทำงานของโปรแกรมต่างๆ.....28

4.1 DI256.C & DIGRAY.C.....28

4.2 LOAD.C.....30

4.3 SAVE.C.....32

4.4 DISP.C.....34

4.5 WRITECH.C.....37

4.6 EDGE.C.....40

4.7 POP2.C.....43

บทที่ 5 สรุปและวิจารณ์.....45

ภาคผนวก.....46

กิตติกรรมประกาศ.....81

หนังสืออ้างอิง.....82

สารบัญรูปภาพ

บทที่ 1

รูป 1.1 แสดงตัวอย่างการใช้ระบบการประมวลผลภาพดิจิตอล.....1

รูป 1.2 แสดงขั้นตอนในการประมวลผลข้อมล.....2

บทที่ 2

รูป 2.1 ความละเอียดของตัวปรับการแสดงผล 4 ชนิด.....7

รูป 2.2 การแอดเดรสพาเล็ทรีจิสเตอร์ในอีจีเอ.....9

รูป 2.3 การแอดเดรสพาเล็ทรีจิสเตอร์ในวีจีเอ.....9

รูป 2.4 การอ้างแอดเดรสของพาเล็ทรีจิสเตอร์ในโหมดต่างๆ.....9

รูป 2.5 การแอดเดรสรีจิสเตอร์สี.....10

รูป 2.6 ขนาดของฟอนต์ต่างๆ.....13

รูป 2.7 การเคลื่อนย้ายข้อมูลไปยังหน่วยความจำแสดงผลโหมด 4,5.....16

รูป 2.8 การเคลื่อนย้ายข้อมูลไปยังหน่วยความจำแสดงผลโหมด 6.....17

รูป 2.9 การเคลื่อนย้ายข้อมูลไปยังหน่วยความจำแสดงผลโหมด 11.....18

รูป 2.10 การเคลื่อนย้ายข้อมูลไปยังหน่วยความจำแสดงผลโหมด F.....19

รูป 2.11 การเคลื่อนย้ายข้อมูลไปยังหน่วยความจำแสดงผลโหมดD,E,10,12...20

รูป 2.12 การเคลื่อนย้ายข้อมูลไปยังหน่วยความจำแสดงผลโหมด 13.....21

บทที่ 3

รูป 3.1 เกรเดียนท์ของ f(x,y) ในทิศทาง r.....23

รูป 3.2 การหาขอบภาพผ่านดังเกรเดียนท์.....23

รูป 3.3 ผลของการหาขอบภาพผ่านโดยดำเนินการโซเบล.....24

รูป 3.4 การหาขอบภาพผ่านตัวคอมพาส.....24

รูป 3.5 ผลของการหาขอบภาพโดยตัวดำเนินการเคิร์ช.....25

รูป 3.6 อนุพันธ์อันดับหนึ่งและสองของการหาขอบภาพ.....26

รูป 3.7 ผลของการหาขอบภาพโดยตัวดำเนินการลาปาซ.....27

สารบัญตาราง

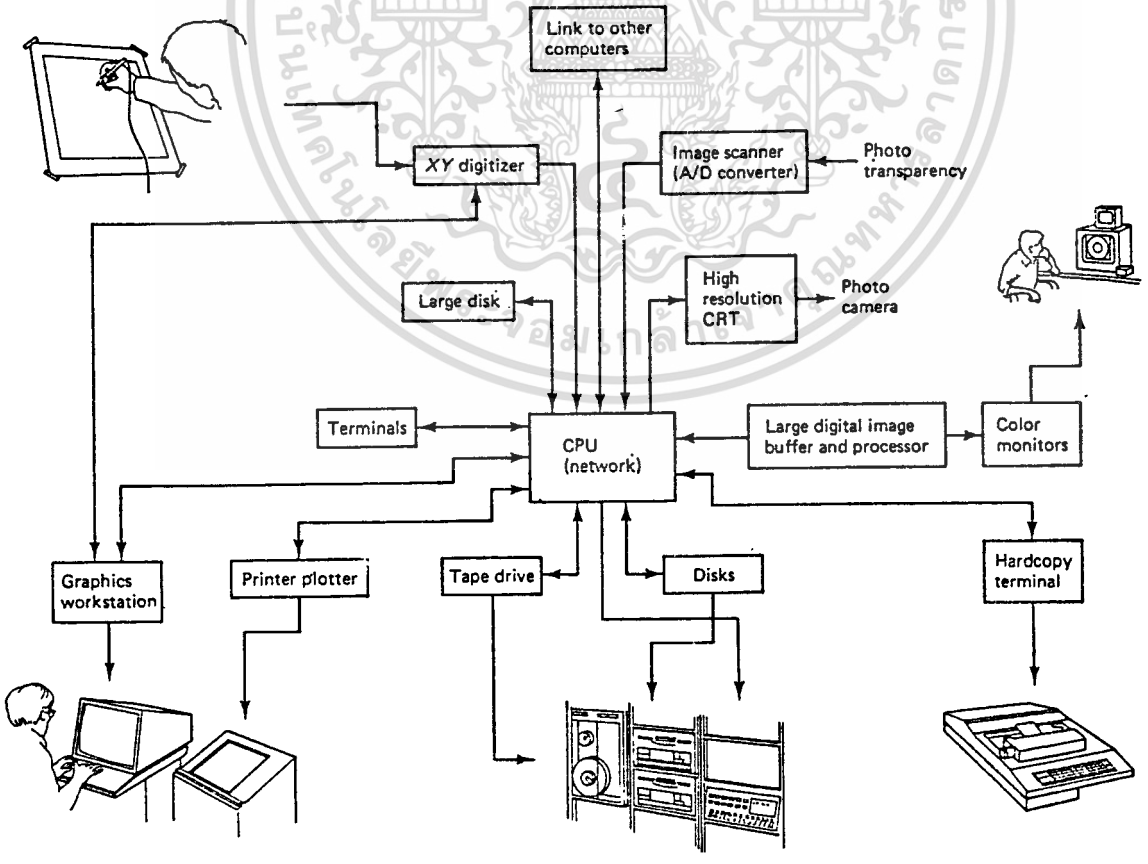
ตาราง 2.1 แอททริบิวต์ของตัวอักษร.....5
ตาราง 2.2 อีจีเอ/วีจีเอรีจิสเตอร์กรุ๊ป.....6
ตาราง 2.3 จำนวนสีที่แสดงได้พร้อมกัน.....7
ตาราง 2.4 จำนวนหน่วยความจำแสดงผล.....8
ตาราง 2.5 โหมดการแสดงผลแบบตัวอักษร.....11
ตาราง 2.6 จำนวนเพจและแอดเดรสเริ่มต้นของแต่ละเพจ.....14
ตาราง 2.7 วิธีการจัดเก็บข้อมูลของแต่ละโหมด.....15
ตาราง 2.8 โหมดต่างๆในโหมดกราฟฟิคส์.....16
ตาราง 2.9 ค่าในพาเล็ทที่กำหนดไว้ในโหมด 6.....17
ตาราง 2.10 ค่าในพาเล็ทที่กำหนดไว้ในโหมด 11.....17
ตาราง 2.11 ค่าในพาเล็ทที่กำหนดไว้ในโหมด D,E,10H,12H.....19
ตาราง 2.12 ค่าในพาเล็ทที่กำหนดไว้ในโหมด 13H.....20

ตาราง 3.1 ตัวดำเนินการเกรเดียนท์ 4 ชนิด.....24
ตาราง 3.2 ตัวดำเนินการคอมพาส 4 ชนิด.....25
ตาราง 3.3 ตัวดำเนินการแบบลาปาซ 4 ชนิด.....27

บทที่ 1
บทนำ

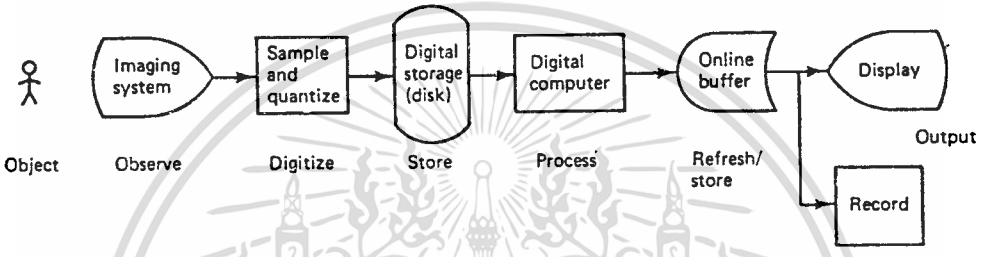
วิทยานิพนธ์เรื่อง " คอมพิวเตอร์กราฟฟิกส์ " (computer graphics) ฉบับนี้ได้กล่าวถึงผลงานต่างๆที่ผู้จัดทำได้ทำขึ้นมาในปีการศึกษา 2532 โดยได้อธิบายถึงทฤษฎีและความรู้ทั่วๆไปเกี่ยวกับอีจีเอ และ วีจีเอ (EGA= Enhanced Graphics Adapter & VGA = Video Graphics Array) , การประมวลผลภาพ (Image Processing) ,การโหลดภาพ, การเซฟภาพ, การหาขอบภาพ, การเชื่อมโยงข้อมูลระหว่างภาพและฐานข้อมูล เป็นต้น ซึ่งในปัจจุบันนี้การประมวลผลภาพและการทำแฟ้มข้อมูลภาพได้เข้ามามีบทบาทในชีวิตประจำวันมากขึ้นเช่น ใช้ในวงการโฆษณา, ใช้ในการจัดเก็บข้อมูลอย่างมีประสิทธิภาพ , ใช้ในงานอุตุนิยมวิทยา, ใช้ในทางการแพทย์ เป็นต้น ดังนั้นในบทนี้จะขอกล่าวถึงการประมวลผลภาพในทางดิจิตอล (Digital Image Processing) ซึ่งเป็นพื้นฐานที่สำคัญเสียก่อน

การประมวลผลภาพในทางดิจิตอล หมายถึงการประมวลผ่ลรูปภาพ 2 มิติโดยดิจิตอลคอมพิวเตอร์ ภาพในทางดิจิตอล (Digital Image) ก็คือ ภาพที่ถูกแทนด้วยเลขจำนวนจริงหรือจำนวนเชิงซ้อน รูป 1.1 แสดงถึงตัวอย่างการใช้ระบบการประมวลผลภาพดิจิตอล



รูป 1.1

ภาพที่ถูก ดิจิไตซ์(Digitize Image) จะถูกนำมาประมวลผลและแสดงบนจอภาพที่มีความละเอียดสูงในการแสดงภาพนั้น ภาพจะถูกเก็บในหน่วยความจำที่สามารถเข้าถึงได้อย่างรวดเร็ว เราสามารถใช้มินิคอมพิวเตอร์หรือไมโครคอมพิวเตอร์ในการติดต่อและควบคุมการ ดิจิไตซ์, การเก็บข้อมูลภาพ, การประมวลผลข้อมูลและกระบวนการแสดงผลผ่านทางโครงข่ายคอมพิวเตอร์ (Computer Network) โดยโปรแกรมจะอินพุทและเอาท์พุทผ่านทางเทอร์มินอล (Terminal) เช่นมอนิเตอร์ (Monitor) หรือ พรินเตอร์ (Printer) เป็นต้น รูป 1.2 แสดงขั้นตอนในการประมวลผลข้อมูล



รูป 1.2

การประมวลผลได้ถูกนำมาประยุกต์อย่างกว้างขวางเช่น ภาพถ่ายจากดาวเทียมหรือจากยานอวกาศ ,การส่งข่าวสารในทางธุรกิจ,การประมวลภาพทางการแพทย์,การประมวลผลภาพจากการมองเห็นของหุ่นยนต์, การประมวลผลภาพที่ได้จากเรดาห์ (Radar) , โซนาร์ (Sonar) และ อะคุสติก (Acoustic) นอกจากนี้ยังสามารถใช้ในงานเครื่องจักรซึ่งตรวจจับการเคลื่อนไหว เป็นต้น

ประโยชน์ของการประมวลผลภาพ

- ภาพถ่ายที่ได้รับจากดาวเทียม เราสามารถนำมาวิเคราะห์และประมวลผลเพื่อหาทรัพยากรธรรมชาติ การเจริญเติบโตของเมือง การทำนายผลผลิตของเกษตรกร การทำนายลักษณะอากาศ เป็นต้น

- ภาพถ่ายที่ได้จาก ยานอวกาศ เราสามารถนำมาวิเคราะห์วัตถุในภาพเพื่อศึกษาระบบสุริยะจักรวาล

- การประยุกต์การส่งภาพข้อมูลเช่น ในการส่งภาพของสถานีโทรทัศน์, การส่งภาพใน ระบบออฟฟิสออโตเมชั่น (Office Automation) การใช้ทีวีวงจรปิดในระบบการรักษาความปลอดภัย

- ภาพถ่ายที่ได้จากการประมวลผลสามารถใช้ในวงการทหาร ตำรวจ หน่วยรักษาความมั่นคงแห่งชาติเพื่อใช้ในการเก็บประวัติบุคคล

- ภาพถ่ายทางการแพทย์สามารถใช้ในการวิเคราะห์โรคมะเร็งหรือ โรคอื่นๆ

-ภาพถ่ายจากระบบเรดาห์หรือโซนาใช้ในระบบนำวิถีของจรวด (Missile System)

เราจะเห็นได้ว่า การประมวลผลภาพนี้มีประโยชน์มากมาย ซึ่งในโครงงานนี้จะเน้นการประมวลผลภาพ ในเรื่องการวิเคราะห์ภาพ (Image Analysis) โดยวิเคราะห์หาขอบภาพ (Edge Detection) เพื่อลดข้อมูลภาพ ซึ่งมีประโยชน์ในด้านการลดพื้นที่ในการเก็บแฟ้มข้อมูลภาพไปได้มากทีเดียว

บทที่ 2
ความรู้ทั่วๆไปเกี่ยวกับอีจีเอ/วีจีเอ

2.1 ลักษณะทั่วๆไปของEGA/VGA

ในปัจจุบันนี้ อีจีเอ และวีจีเอ ได้เริ่มเป็นที่นิยมกันมากขึ้น ทั้งนี้เนื่องจากว่ามีราคาไม่สูงนัก, มีความเร็วสูง, มีหน่วยความจำความจุสูง,กะทัดรัดและง่ายต่อการรักษา

คำว่า "EGA/VGA boards" จะหมายความถึงไบออส (BIOS) ของมันด้วย ซึ่งทำให้สามารถใช้ได้กับซอฟท์แวร์ (software) อื่น ๆ ได้ ทั้งนี้จะรวมไปถึง เวิร์ดโปรเซสเซอร์ (word processors),การจัดการฐานข้อมูล (data base managers),สเปรดชีท (spreadsheet) และ ยูทิลิตี้ (UTILITY) อื่น ๆ

ในปัจจุบันนี้ การ์ดอีจีเอ/วีจีเอ (EGA/VGA cards) ส่วนใหญ่จะพัฒนามาตรฐานต่างๆดีขึ้นมามาก เช่น ความละเอียด,สี, หน่วยความจำ, ตัวอักษร (character fonts) ฯลฯ แต่มีข้อเสียคือ ไม่ได้ถูกควบคุมโดยมาตรฐานใดๆ ทำให้ต้องใช้ตัวขับ (drivers) ในการใช้ลักษณะเด่นที่เพิ่มขึ้นเหล่านี้กับซอฟท์แวร์ ที่ได้กล่าวมาแล้ว ซึ่งก่อให้เกิดความไม่สะดวกแก่ผู้ใช้ในกรณีที่ต้องการใช้ซอฟท์แวร์ หลายๆตัว เพราะว่า แต่ละตัวอาจไม่ใช้ตัวขับตัวเดียวกันก็ได้

2.2 หน่วยความจำแสดงผล (display memory)

ใน อีจีเอ/วีจีเอ นั้น ผู้ใช้สามารถอ่านหรือเขียนไปยัง หน่วยความจำแสดงผล โดยไม่ต้องรอช่วงเวลาของการ รีเทรส แนวนอนหรือแนวดิ่ง ซึ่งจะทำให้กระบวนการทาง กราฟฟิกส์ (graphics) ของ อีจีเอ/วีจีเอ มีความเร็วสูงกว่าอแด็ปเตอร์แสดงผล (display adapters) อื่นๆ นอกจากนี้ หน่วยความจำยังถูกจัดในลักษณะตรงไปตรงมา ซึ่งหากผู้ใช้เข้าใจในการอ้างแอดเดรส (addressing) แล้ว ก็จะสามารถวาดเส้นและรูปต่างๆไปยังจอได้โดยตรง

หน่วยความจำแสดงผล ประกอบด้วย หน่วยความจำ 256 เคไบต์ (Kbytes) ติดต่อกับโปรเซสเซอร์ (processor) ผ่านทาง 8 บิทบัสข้อมูล (bit data bus) สามารถแสดงความละเอียดได้หลายอย่าง ขึ้นอยู่กับ โหมดการแสดงผล (display mode) ที่ใช้ และสามารถใช้แพคฟอร์แมท (packed format) หรือ บิทเพลนฟอร์แมท (bit-plane format) ก็ได้ ขึ้นอยู่กับ โหมดการแสดงผล ที่ใช้งานในขณะนั้น

2.3 โหมดตัวอักษร (Alphanumeric Modes)

หน่วยความจำแสดงผล สามารถใช้คอมแพทติเบิล (compatible) กับ เอ็มดีเอ (MDA) และ ซีจีเอ (CGA) ได้ ตัวอักษรแต่ละตัวถูกแทนใน หน่วยความจำแสดงผล เพียง 2 ไบต์ ไบต์แรกคือ รหัสแอสกี (ASCII character code) ซึ่งเป็นตัวบอกให้ทราบว่า ตัวอักษร (character) ตัวใดที่ต้องการจะแสดง ไบต์ที่สองคือ แอททริบิวต์ (attribute) ซึ่งเป็นตัวบอกสีของ ตัวอักษร

ในหน่วยความจำ ไบต์ที่แสดงตัวอักษร และ แอททริบิวต์ ถูกเก็บในลักษณะเรียงติดกันไป การอ้างแอดเดรส ที่เจาะจงลงไป จะทำให้ ตัวอักษร และ แอททริบิวต์ ถูกเคลื่อนย้ายไปยัง

หน่วยความจำแสดงผล ในลักษณะที่ยังเรียงติดกันด้วย ตัวอักษร สามารถแสดงโดยโมโนโครม (monochrome) หรือ แอททริบิวต์สี ดังแสดงในตารางที่ 2.1

ตารางที่ 2.1 แอททริบิวต์ของตัวอักษร

โหมด	แอททริบิวต์
Mono	Normal,Intensified,Reversed,Blinking,Underlined
Color	16 foreground,16 background colors
Color	16 foreground,8 background,blinking
Color	8 foreground,8 background,512 characters,blinking

2.4 <u>โหมดกราฟฟิคส์ (Graphics Modes)</u>

เนื่องจาก หน่วยความจำแสดงผล ถูกจัดในลักษณะที่บางโหมดสามารถใช้ร่วมกับซีจีเอ (CGA) ได้ ดังนั้นจึงทำให้สามารถใช้กราฟฟิคส์ของซีจีเอได้ด้วย โหมดกราฟฟิกส์ แบ่งได้เป็น 2 โหมดใหญ่ๆ คือ โหมดโมโนโครม และ โหมดสี

ในโหมดโมโนโครม มี 2 แบบ คือ 2 บิทต่อ 1 จุดภาพ และ 1 บิทต่อ 1 จุดภาพ แบบแรกใช้กับโปรแกรมในเฮอร์คิวลิส (Hercules) แบบหลังเรียกว่า โหมด 2 สี (two-color modes) เพราะว่าโหมดนี้สามารถแสดงได้เพียง 2 สี จึงยังถูกจัดให้อยู่ในโหมดโมโนโครม สำหรับ 2 สีที่แสดงนี้ไม่จำเป็นต้องเป็นสีขาวและสีดำ

ในโหมดสี แบ่งเป็น 3 แบบ คือ 2,4,และ 8 บิทต่อ 1 จุดภาพ ในโหมด 2 บิทต่อ 1จุดภาพ เลียนแบบซีจีเอ แสดงได้ 4สีต่อภาพในเวลาเดียวกัน ในโหมด 4 บิทต่อ 1จุดภาพ แสดงได้ 16 สีพร้อมกัน และโหมด 8 บิทต่อ 1จุดภาพ แสดงได้ 256 สีพร้อมกัน

2.5 <u>รีจิสเตอร์ควบคุม (Control Registers)</u>

รีจิสเตอร์ ถูกแบ่งออกเป็น 5 กลุ่ม ดัง ตาราง ที่ 2.2.

รีจิสเตอร์ แต่ละตัวมีขนาด 1 ไบต์ รีจิสเตอร์บางตัวถูกแบ่งย่อยเป็นฟิล์ด (field) ดังนั้น บ่อยครั้งที่บิทแต่ละบิทภายใน รีจิสเตอร์ เดียวกันจะควบคุมฟังก์ชันที่ต่างกัน และบางฟังก์ชันก็ถูกควบคุมโดย รีจิสเตอร์ ที่มากกว่า 1 ตัว

รีจิสเตอร์ ถูกเข้าถึงโดยผ่านการแอดเดรสไอโอพอร์ท (I/O Port) แต่ละกลุ่มของรีจิสเตอร์ ที่แสดงในตารางที่2.2 ถูกแทนด้วย รีจิสเตอร์ 2 ตัวในไอโอพอร์ท รีจิสเตอร์ของ อีจีเอ/วีจีเอ ถูกชี้โดยการแอดเดรสผ่านอินเด็กซ์รีจิสเตอร์ (index register หรือ address register) และรีจิสเตอร์ข้อมูล (data register)

ตาราง 2.2 กลุ่มรีจิสเตอร์ของ อีจีเอ/วีจีเอ

จำนวนรีจิสเตอร์	ชื่อกลุ่ม
4	General registers
5	Sequencer registers
28	CRT controller registers
9	Graphics controller registers
22	Attribute controller
5	Color registers

2.6 โหมดการแสดงผล (Display Modes)

โหมดการแสดงผล แบ่งได้เป็น 2 โหมด คือ โหมดตัวอักษรและโหมดกราฟฟิค (all-points-addressable) ซึ่งทั้ง 2 โหมดมีความแตกต่างกันดังที่ได้อธิบายมาแล้ว นอกจากนี้ผู้ใช้ยังสามารถออกแบบโหมดการทำงานขึ้นมาเองได้ โดยการใช้ รีจิสเตอร์ แต่วิธีนี้มีข้อเสียคือ ความซับซ้อนและความยุ่งยาก ซึ่งหากเกิดข้อผิดพลาดขึ้น อาจทำความเสียหายให้กับจอ หรือ ตัวปรับ (adapter) ได้ วิธีที่สะดวกและมีประสิทธิภาพ คือ การใช้ไบออส

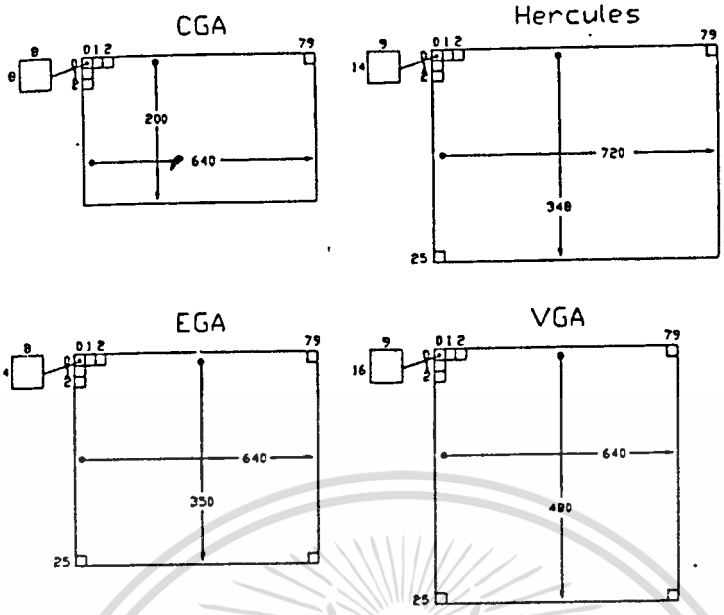
2.7 ความละเอียดของอีจีเอและวีจีเอ (EGA & VGA Resolution)

ความละเอียด ของ อีจีเอ/วีจีเอ ในโหมดความละเอียดต่ำ คือ 320 x 200 จุดภาพ (pixels) ในโหมดความละเอียดสูง คือ 640 x 350 จุดภาพ และวีจีเอ ยังสามารถให้ความละเอียด ที่เพิ่มขึ้นถึง 640 x 480 จุดภาพ ความละเอียดของตัวปรับการแสดงผล ที่เป็นที่นิยมกันทั้ง 4 ชนิด แสดงในรูปที่ 2.1

2.8 ความละเอียดของสี (Color Resolution)

โหมดการแสดงผลหลายโหมดใน อีจีเอ/วีจีเอ สามารถแสดงได้ 16 สีพร้อมกัน ซึ่งเลือกได้จาก 64 สีใน อีจีเอ และ 256K สีใน วีจีเอ นอกจากนี้ โหมด 13Hex ใน วีจีเอ สามารถแสดงได้ถึง 256 สีพร้อมกันจาก 256K สี(262,144สี)

จำนวนของสีที่แสดงได้พร้อมกัน มีความสัมพันธ์กับจำนวนบิทที่เกี่ยวข้องกับแต่ละจุดภาพใน หน่วยความจำแสดงผล บิทยิ่งมาก สียิ่งมาก ดังแสดงในตารางที่ 2.3



รูป 2.1 ความละเอียดของตัวปรับแสดงผล 4 ชนิด

ตาราง 2.3 จำนวนของสีที่แสดงได้พร้อมกัน

บิท/จุดภาพ	จำนวนสี
1	2
2	4
4	16
8	256

ค่าของจำนวนสีจะเป็นค่าที่ได้จาก 2 ยกกำลังด้วยค่าจำนวนบิท/จุดภาพ. หน่วยความจำแสดงผล ถูกจัดเป็นไบต์ๆ เพื่อที่จะใช้หน่วยความจำให้เกิดประโยชน์ได้เต็มที่ จึงควรบรรจุ จุดภาพ เป็นคู่ลงไปในไบต์ ความละเอียดซึ่งเกี่ยวข้องกับจำนวนบิท/จุดภาพ จะกำหนดจำนวนของ หน่วยความจำแสดงผล ทั้งหมด ดังแสดงในตารางที่ 2.4

ตาราง 2.4 จำนวนหน่วยความจำแสดงผล

ความละเอียด	บิท/จุดภาพ	จำนวนหน่วยความจำ
320 x 200	1	8,000 ไบต์
320 x 200	2	16,000 ไบต์
320 x 200	8	64,000 ไบต์
640 x 200	2	32,000 ไบต์
640 x 350	1	28,000 ไบต์
640 x 350	2	56,000 ไบต์
640 x 350	4	112,000 ไบต์
640 x 480	1	38,400 ไบต์
640 x 480	4	153,600 ไบต์

2.9 พาเล็ทสีและรีจิสเตอร์สี (COLOR PALETTE AND COLOR REGISTER)

พาเล็ทสี (Color Palette)

พาเล็ทสี เป็นตัวที่ทำหน้าที่เปลี่ยนข้อมูลใน หน่วยความจำแสดงผล ให้เป็นสี ใน วีจีเอ พาเล็ทสี จะมี รีจิสเตอร์สี เพิ่มเข้ามาเพื่อเพิ่มประสิทธิภาพ

พาเล็ทสี เปลี่ยนสีได้อย่างรวดเร็ว เพราะจะทำการเปลี่ยนเฉพาะข้อมูลที่ต้องการแก้ไขเท่านั้น เช่น ภาพใน อีจีเอ/วีจีเอ กินเนื้อที่ใน หน่วยความจำแสดงผล 256Kไบต์ ในโหมด 16 สี จะแสดงสีได้พร้อมกัน 16 สี ถ้าไม่มีพาเล็ทสีแล้ว ข้อมูลทั้ง 256Kไบต์ ใน หน่วยความจำแสดงผล จะถูกแก้ไขทั้งหมด แต่เมื่อมี พาเล็ทสี แล้ว ข้อมูลเพียง 16 ไบต์เท่านั้นที่ถูกแก้ไข

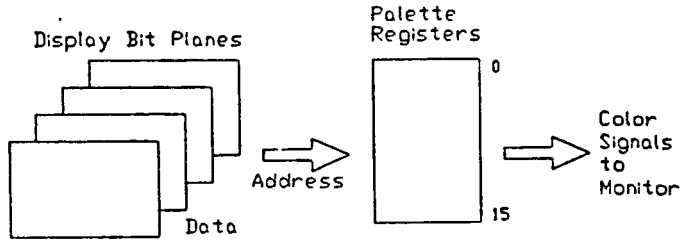
ประโยชน์อีกอย่างของการใช้พาเล็ทสี คือ จำนวนสีที่สามารถแสดงได้จะมากกว่าจำนวนสีที่สามารถแสดงได้พร้อมกัน เช่น อีจีเอ สามารถแสดงได้ 16 สีพร้อมกันจากทั้งหมด 64 สี แต่ละสีในพาเล็ทรีจิสเตอร์ ถูกแทนด้วยคำขนาด 6 บิท ใน วีจีเอ มีโหมด 256 สี ซึ่งแสดงได้ 256 สีในเวลาเดียวกัน แต่ละสีถูกแทนด้วยคำขนาด 18 บิทภายในรีจิสเตอร์สี ซึ่งมีถึง 256K สีให้เลือก

พาเล็ทรีจิสเตอร์ (Palette registers)

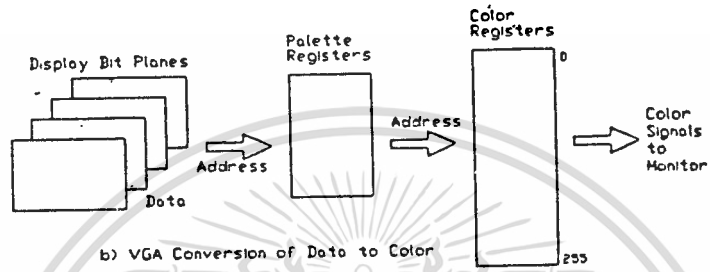
พาเล็ทสี ประกอบด้วย พาเล็ทรีจิสเตอร์ 16 ตัว ข้อมูลใน บิทเพลน ซึ่งตรงกับแต่ละจุดภาพ จะเป็นตัวชี้พาเล็ทรีจิสเตอร์.

ใน อีจีเอ :พาเล็ทรีจิสเตอร์ ที่ถูกชี้จะส่งรหัสสีไปยัง มอนิเตอร์ ดังรูป 2.2

ใน วีจีเอ :พาเล็ทรีจิสเตอร์ ที่ถูกเลือกจะสร้างแอดเดรสซึ่งจะชี้ไปยังรีจิสเตอร์สี ซึ่งรีจิสเตอร์สี จะส่งรหัสสีไปยังมอนิเตอร์ ดังรูป 2.3

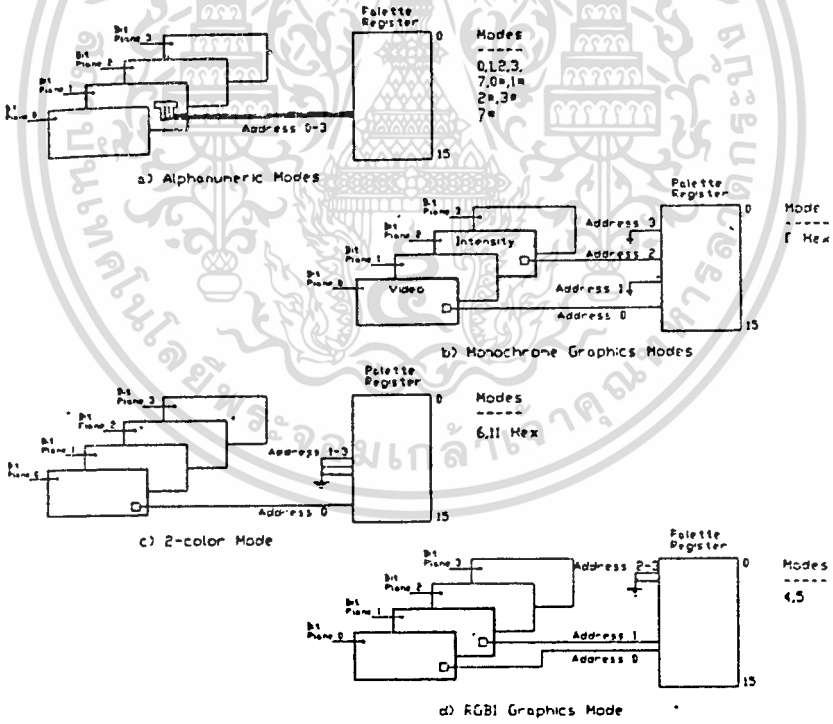


รูป 2.2



b) VGA Conversion of Data to Color

รูป 2.3

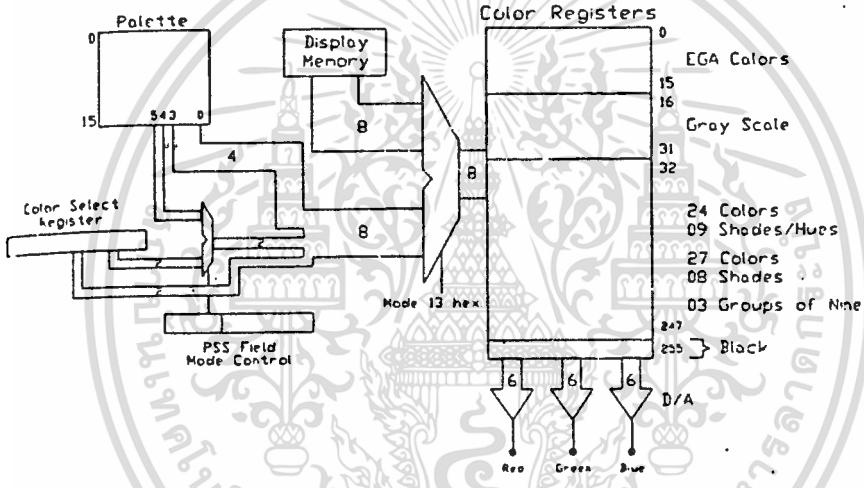


รูป 2.4

รีจิสเตอร์สี (Color Register)

ใน วีจีเอ จะมี 256 รีจิสเตอร์สี แต่ละจุดภาพหรือ ตัวอักษร ประกอบด้วยข้อมูล 8 บิท ซึ่งใช้แอดเดรส 16 พาเล็ทรีจิสเตอร์. เอาท์พุทของ พาเล็ทรีจิสเตอร์ ร่วมกับ ข้อมูลจาก หน่วยความจำแสดงผล จะใช้แอดเดรสรีจิสเตอร์สี ซึ่ง รีจิสเตอร์สี จะส่งรหัสสีไปยัง ดิจิตอลทู อนาล็อก คอนเวิตเตอร์ (digital-to-analog converters หรือ DAC) โดยที่เอาท์พุตของ DAC จะถูกส่งไปยังจอต่อไป รีจิสเตอร์สี แต่ละตัวจะประกอบด้วย 3 ส่วน คือ แดง, เขียว และ น้ำเงิน แต่ละส่วนกว้าง 6 บิท

สำหรับ พาเล็ทรีจิสเตอร์ในวีจีเอนั้นมีเพียงเพื่อให้สามารถคอมแพทกับ อีจีเอ เมื่ออยู่ในโหมด 13 Hex แล้ว รีจิสเตอร์สี เพียงอย่างเดียวก็เพียงพอ ซึ่งการแอดเดรส รีจิสเตอร์สี แสดงดังรูป 2.5



รูป 2.5

2.10 การประมวลผลทางตัวอักษร (ALPHANUMERIC PROCESSING)

ในโปรแกรมจำพวกโปรแกรมประยุกต์ (APPLICATION PROGRAM) ต่างๆเช่น เวิร์ดโปรเซสเซอร์ (WORD PROCESSOR), สเปรดชีท (SPREADSEET), ฐานข้อมูล(DATABASE), หรือแม้แต่ดอส (DOS) เองก็ตาม มักนิยมใช้โหมดตัวอักษรในการแสดงผลตัวอักษรมากกว่าโหมดกราฟฟิคส์ เหตุผลที่สำคัญก็คือ ในโหมดตัวอักษร จะแสดงตัวอักษรได้เร็วกว่า โหมดกราฟฟิคส์มาก โดยในโหมดกราฟฟิคส์ หากเราจะเขียนตัวอักษรสักตัวหนึ่ง เราจะต้องกำหนดจุดทุกจุดบนจอให้ได้รูป ตามแบบตัวอักษร (ดูเรื่อง GRAPHIC PROCESSING)แต่ในที่นี้จะแสดงคร่าวๆว่าการจะเขียนตัวอักษรขนาด 8*14 นั้นจะใช้จำนวนไบต์ มากกว่าในโหมดตัวอักษรเท่าใด

โหมดกราฟฟิคส์ (ใน โหมด 16 สี)จะใช้ 4 บิท ในการบอก แอททริบิวต์ ดังนั้น ฟอนต์ แบบ 8*14 จะใช้ 14 ไบต์*4 = 56 ไบต์

โหมดตัวอักษร ใช้ 2 ไบต์ คือ รหัสตัวอักษร และ แอททริบิวต์ จะเห็นว่าใน โหมดกราฟฟิคส์ นั้นต้องเขียนข้อมูลมากกว่าในโหมดตัวอักษร 28 เท่า

เหตุที่ในโหมดตัวอักษรเร็วกว่า โหมดกราฟฟิคส์ (ในการแสดงตัวอักษร)ก็คือในโหมดตัวอักษร จะโหลดชุดของตัวอักษร (CHARACTER FONT)ลงบนหน่วยความจำแสดงผล บิทเพลน 2 หรืออาจเกินมาถึง บิทเพลน 3 เวลาเราเขียนรหัสตัวอักษร ซึ่งก็คือ รหัสแอสกีนั่นเอง จะมีกลไกทางฮาร์ดแวร์ ทำให้รหัสนั้นลงไปเก็บที่ หน่วยความจำแสดงผล บิทเพลน 0 และเก็บ แอททริบิวต์ไบต์ไว้ที่ บิทเพลน 1 จากนั้นก็จะนำข้อมูลจาก บิทเพลน 0 ไปค้นหาว่าเป็นตัวอักษรใดที่บิทเพลน 2,3 แล้วนำมารวมกับ แอททริบิวต์ไบต์ ซึ่งเป็นตัวบอกสีของฟอร์กราวด์ (FOREGROUND) และ แบ็คกราวด์ (BACKGROUND) แล้วนำมาแสดงผลบนจอภาพ เช่น สมมติว่าเราต้องการเขียนตัวอักษร "A" ซึ่งมี รหัสแอสกี 65 ดังนั้นเมื่อเราป้อนค่า 65 ลงบน บิทเพลน 0 มันจะนำไปค้นหาที่ บิทเพลน 2,3 ฟอนต์ ของตัว "A" จะอยู่แอดเดรส ที่

อักษรตัวแรกของ ฟอนต์ + (65 * จำนวนแถว)

ตาราง 2.5 โหมดการแสดงผลแบบตัวอักษร(ALPHANUMERIC DISPLAY MODE)

โหมด(HEX)	แถว*คอลัมน์	ขนาด	จำนวนสี	#เพจ	ความละเอียด
0,1	40*25	8*8	16	8	320*200
0*,1*	40*25	8*14	16	8	320*350
0+,1+	40*25	9*16	16	8	360*400
2,3	80*25	8*8	16	8	640*200
2*,3*	80*25	8*14	16	8	640*350
2+,3+	80*25	9*16	16	8	720*400
7	80*25	9*14	bw	8	720*350
7+	80*25	9*16	bw	8	720*400

ข้อสังเกตุ เครื่องหมาย * และ + หมายถึง กาจ์ด (CARD) ที่ต่ออยู่กับ มอนิเตอร์ที่มีความละเอียดสูง

โหมดตัวอักษรแบบสี (RGBI Alphanumeric Mode)

โหมด 0 และ โหมด 1

เป็นโหมดการแสดงผลตัวอักษรขนาด 40 คอลัมน์ 16 สี มีความละเอียด 320 จุดภาพ/เส้นสแกน โดยมี ตัวอักษรที่กำหนดไว้ขนาด 8*8 เมื่อใช้ 200 เส้นสแกนแนวดิ่ง และมี ตัวอักษรที่กำหนดไว้ขนาด 8*8 เมื่อใช้ 350 เส้นสแกนแนวดิ่ง และมีความละเอียด 360 จุดภาพ/เส้นสแกน โดยมี ตัวอักษรที่กำหนดไว้ขนาด 9*16 เมื่อใช้ 400 เส้นสแกนแนวดิ่ง

มี แอดเดรส เริ่มต้นที่ B8000 HEX

โหมด 2 และ โหมด 3

เป็นโหมดการแสดงผลตัวอักษร 80 คอลัมน์ 16 สี มีความละเอียด 640 จุดภาพ/เส้นสแกน

โดยที่ 200 เส้นสแกนแนวดิ่ง จะมี ตัวอักษรที่กำหนดไว้ขนาด 8*8

ที่ 350 เส้นสแกนแนวดิ่ง จะมี ตัวอักษรที่กำหนดไว้ขนาด 8*14 และความละเอียด 720*400 จุดภาพ โดยมี ตัวอักษรที่กำหนดไว้ขนาด 9*16

มี แอดเดรส เริ่มต้นที่ B8000 HEX

โหมดตัวอักษรแบบขาวดำ (Monochrome Alphanumeric Mode]โหมด 7)

ในโหมด7นี้ รหัสของตัวอักษรแต่ละตัวจะอยู่ใน แอททริบิวต์ไบต์ ของ ตัวอักษร ซึ่ง แอททริบิวต์ ไบต์ อยู่ใน บิทเพลน2 ในโหมดนี้ แอททริบิวต์ ไบต์ สามารถเข้าถึง พาเล็ทรีจิสเตอร์ ได้ทั้ง 16 ตัว แต่มีค่าเพียง 4 ค่าเท่านั้น ทั้งนี้เนื่องจากว่ารหัสจะอยู่ใน 2 บิทเท่านั้นคือ บิท 3 และ บิท 4 บิท 0-2, 5-7 ไม่มีผล บิท 3 จะเป็นบิทแสดงสัญญาณวิดีโอ (video) ส่วน บิท 4 เป็นบิทแสดงความเข้มของสัญญาณ

โหมด 7 เป็นโหมดการแสดงผลตัวอักษร 80 คอลัมน์ สี ขาว ดำ ซึ่ง คอมแพททิเบิล กับ โมโนโครม (MDA) มีความละเอียด 720*350 โดยมี ตัวอักษรที่กำหนดไว้ขนาด 9*14 และมีความละเอียด 720*400 โดยมี ตัวอักษรที่กำหนดไว้ขนาด 9*16

มี แอดเดรส เริ่มต้นที่ B8000 HEX

ฟอนต์ของตัวอักษร (CHARACTER FONT)

ขนาดของตัวอักษรใน ฟอนต์ นั้นมีตั้งแต่ 1-32 แถว ซึ่งสามารถกำหนดได้โดยการกำหนดจำนวนเส้นสแกนสูงสุด ในซีอาร์ทีคอนโทรลเลอร์รีจิสเตอร์ (CRT CONTROLLER REGISTER) แต่ฟอนต์ที่พบในรอมไบออส (ROM BIOS) นั้นมีอยู่ 5 แบบคือ 8*8,8*14,8*16,9*14,9*16 ซึ่งมีลักษณะดังรูป 2.6 การคำนวณหาตำแหน่งของ ฟอนต์ สามารถคำนวณได้จากสูตร

CHAR PATTERN ADDR = CHAR BASE ADDR + (ASCII CODE * #BYTE/CHAR)

เช่นตัวอักษร "A" ในตัวอย่างที่แล้ว สมมติว่าอยู่ใน ฟอนต์ 8*8 ซึ่งอยู่ที่หน่วยความจำแสดงผล บิทเพลน 2 ที่ ออฟเซทแอดเดรส 2000 HEX

ดังนั้นที่ แอดเดรส = (2000HEX)+(65*8DEC)

= 2000HEX + 520DEC = 2000HEX + 208HEX

= 2208HEX

จะมีค่าเหล่านี้อยู่

แอดเดรส 2208 -----> 30HEX

แอดเดรส 2209 -----> 48HEX

แอดเดรส 2210 -----> 84HEX

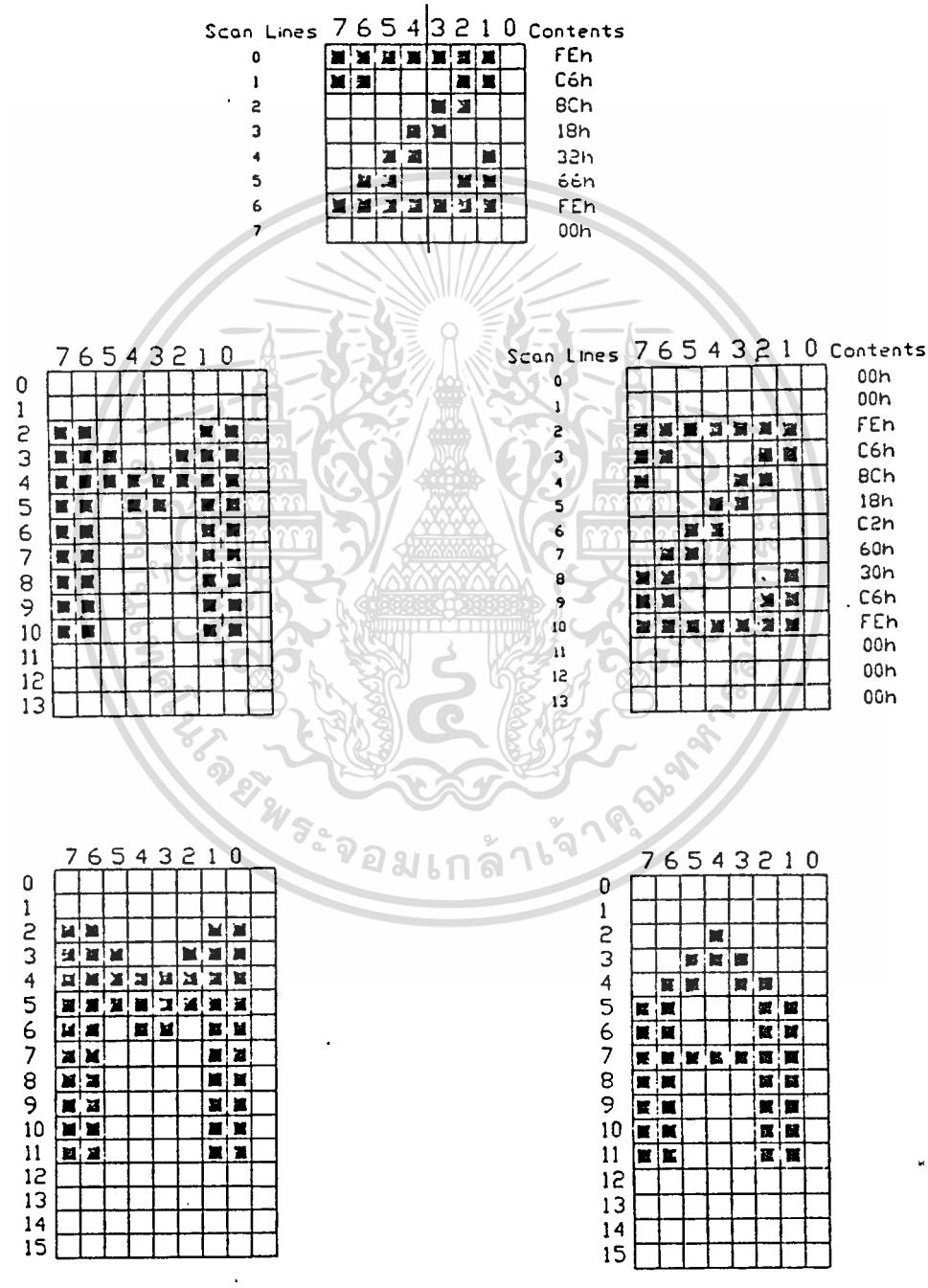
แอดเดรส 2211 -----> 84HEX

แอดเดรส 2212 -----> FCHEX

แอดเดรส 2213 -----> 84HEX

แอดเดรส 2214 -----> 84HEX

แอดเดรส 2215 -----> 00HEX



รูป 2.6 แสดงฟอนต์ตัวอักษร

2.11 การประมวลผลทางกราฟฟิคส์ (GRAPHICS PROCESSING)

ในปัจจุบันคอมพิวเตอร์กราฟฟิคส์ (COMPUTER GRAPHIC) ได้มีการพัฒนาไปเป็นอย่างมาก เนื่องจาก รูปภาพสามารถสื่อความเข้าใจได้อย่างรวดเร็ว เช่น ถ้าต้องพิจารณาข้อมูลที่เป็นตัวเลขมากมายก็จะทำให้ยากต่อการเข้าใจ หากนำมาพล็อตเป็นกราฟแล้ว จะสามารถหาข้อสรุปเกี่ยวกับจำนวนเหล่านั้นได้อย่างรวดเร็ว เช่น อัตราการเพิ่ม,ลด นอกจากนี้ยังง่ายต่อการนำเสนออีกด้วย

ทางด้าน ฮาร์ดแวร์ของการแสดงผลก็ได้มีการพัฒนาเช่นเดียวกับ ซอฟท์แวร์ ในปัจจุบัน ภาคแสดงผลของคอมพิวเตอร์ มีขีดความสามารถใกล้เคียงกับเวิร์คสเตชั่น (WORKSTATION) เช่น วีจีเอ(VIDEO GRAPHIC ARRAY) นั้นสามารถแสดงสีได้คราวละ 256 สี จากสีที่ให้เลือกทั้งหมด 256K สีที่ความละเอียด 320*200 จุดภาพ

ในโหมดตัวอักษรนั้น ใช้ 2 ไบต์ ในการแทนตัวอักษร 1 ตัวคือ รหัสแอสกี และ แอททริบิวต์ไบต์ ส่วนโหมดกราฟฟิคส์นั้น ตำแหน่งในหน่วยความจำทุกเพลน จะแทนจุดบนจอภาพ โดยอาจเป็น 1 บิท/จุดภาพ, 2 บิท/จุดภาพ, 4 บิท/จุดภาพ, 8 บิท/จุดภาพ ขึ้นอยู่กับจำนวนสีที่ต้องการแสดงผล ถ้าหากต้องการให้มีสีหลายๆสี ก็ต้องใช้จำนวน บิท/จุดภาพ มาก ซึ่งจะทำให้ความละเอียดของภาพลดลง

หน่วยความจำแสดงผลของวีจีเอ ประกอบด้วย หน่วยความจำอ่านเขียนแบบไดนามิค (DYNAMIC READ/WRITE MEMORY) ซึ่งแบ่งออกเป็น 4 ส่วน(4 บิทเพลน หรือ 4 bit map) แต่ละส่วนจะมีขนาดตั้งแต่ 16K ไบต์ ไปจนถึง 64 K ไบต์ ขึ้นอยู่กับโหมดการแสดงผล โหมดที่ใช้หน่วยความจำจะสามารถมีได้หลายเพจ ในตาราง2.6 แสดงจำนวนเพจ และ แอดเดรสเริ่มต้น ของแต่ละโหมด

ตาราง 2.6

เพจ	โหมด4,5H	โหมด D	โหมด E	โหมด 10H	โหมด 11H	โหมด 12H	โหมด13H
0	B8000	A0000	A0000	A0000	A0000	A0000	A0000
1	-	A2000	A4000	A8000	-	-	-
2	-	A4000	A8000	-	-	-	-
3	-	A6000	AC000	-	-	-	-
4	-	A8000	-	-	-	-	-
5	-	AA000	-	-	-	-	-
6	-	AC000	-	-	-	-	-
7	-	AE000	-	-	-	-	-

มีเทคนิค 2 ประการ ในการจัดเก็บข้อมูลลง หน่วยความจำแสดงผล คือ บิทเพลน และ แพคดิสเพลย์ฟอร์แมท (Packed Display Format)

-<u>บิทเพลน</u> เป็นการเก็บข้อมูลของ จุดภาพ โดยแบ่งเป็นเพลน เช่น ในโหมด 16สี ใช้ 4 บิท/จุดภาพ แต่ละบิท นี้ จะถูกจัดเก็บให้อยู่บนเพลนต่างๆกัน คือ บิท 0 อยู่บน เพลน 0 ,บิท 1,2,3 อยู่บน เพลน 1, 2, 3 ตามลำดับ

-<u>แพคดิสเพลย์ฟอร์แมท</u> เป็นการเก็บข้อมูลของ จุดภาพ ลงในเพลนเดียวกัน

ตาราง 2.7 แสดงการใช้วิธีการจัดเก็บข้อมูลของจุดภาพในแต่ละโหมด

โหมด (HEX)	รูปแบบ	จำนวนคอลัมน์	จุดภาพ/ไบต์
4,5	แพค	4	4
6	แพค	2	8
F	บิทเพลน	2	8
11	แพค	2	8
D	บิทเพลน	16	8
E,10	บิทเพลน	16	8
12	บิทเพลน	16	8
13	แพค	256	1

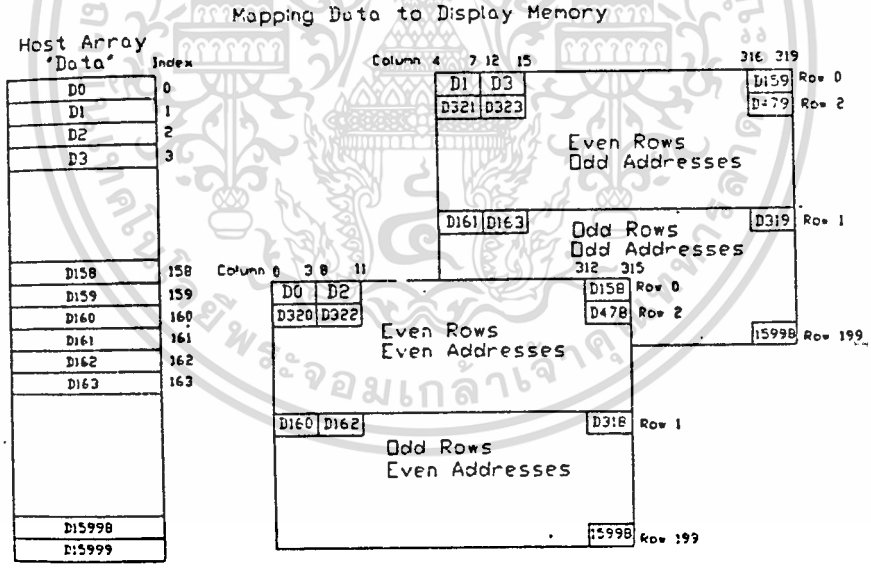
<u>โหมดกราฟฟิคส์สี (RGBI Graphics Modes]โหมด 4,5)</u>

โหมดกราฟฟิคส์สี คือ โหมด 4 และ 5 แต่ละ จุดภาพ ถูกแทนด้วย 2 บิทแอททริบิวต์สี โหมดนี้สามารถใช้ได้กับซีจีเอ ด้วย

ในโหมด 4 และ 5 แอททริบิวต์บิท ทั้ง 2 บิทอยู่ติดกันในข้อมูลไบต์เดียวกัน อุปกรณ์ทางฮาร์ดแวร์ จะควบคุมทางแฟลกคู่/คี่ (odd/even flag) คือ ข้อมูลของไบต์คู่จะถูกส่งไปยังบิทเพลน 0 และข้อมูลไบต์คี่จะถูกส่งไปยัง บิทเพลน 1 ดังนั้นข้อมูลที่ถูกส่งเข้าไปยัง อีจีเอ/วีจีเอ เมื่ออยู่ในโหมด 4 หรือ 5 จะถูกแบ่งแยกไปที่ บิทเพลน 0 และ 1 ทั้งสองโหมดนี้เป็นการแสดงผลขนาด 320*200 จุดภาพ 4 สี ดังนั้นใน 1 จุดภาพ จึงต้องการ 2 บิท จึงจะแสดงได้ 4 สี โดยจะใช้ บิทเพลน 0,1 ในการเก็บข้อมูล ดังรูป 2.7

ตาราง 2.8 แสดงโหมดต่างๆ ในโหมดกราฟฟิคส์

โหมด(HEX)	จำนวนหน้า	แอดเดรส	จำนวนสี	ความละเอียด
4	1	B8000	4/256K	320*200
5	1	B8000	4/256K	320*200
6	1	B8000	2/256K	640*200
D	8	A0000	16/256K	320*200
E	4	A0000	16/256K	640*200
F	2	A0000	bw	640*350
10	1	A0000	16/256K	640*350
11	1	A0000	2/256K	640*480
12	1	A0000	16/256K	640*480
13	1	A0000	256/256K	320*200



รูป 2.7

โหมด 2 สี (Two-Color Modes)

สามารถแบ่งย่อยได้เป็น 2 โหมด ดังนี้

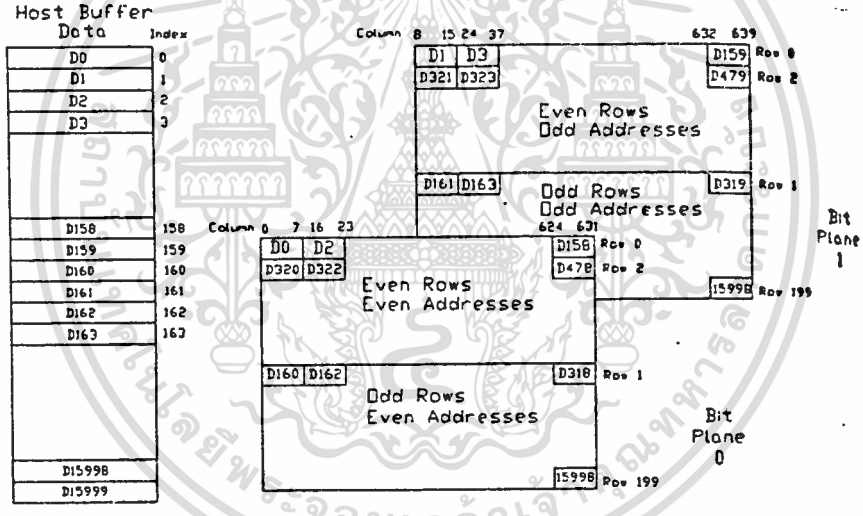
ซีจีเอ็กคัลเลอร์อีมูเลชั่น (CGA Two-Color Emulation]โหมด6)

ในโหมด 6 ของ อีจีเอ/วีจีเอ 1 บิทต่อ 1 จุดภาพ จะเลือก 1 ใน 2ของพาเล็ท ค่าที่โหลดลงไปใน พาเล็ทรีจิสเตอร์ แสดงในตาราง 2.9

ตาราง 2.9 ค่าในพาเล็ทที่กำหนดไว้ในโหมด 6 Hex

รีจิสเตอร์	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ค่า	0	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17

ใน หน่วยความจำแสดงผล แต่ละ ไบต์ จะประกอบด้วย 8 จุดภาพที่ต่อเนื่องกัน วิธีการต่างๆจะเหมือนกับในโหมดกราฟิคส์สี (โหมด 4 และ 5) ยกเว้นแต่ว่า แสดงได้ 2 สี แทนที่จะเป็น 4 สี ในโหมด 6 นี้ บิทเพลน 0 และ 1 ถูกใช้เก็บข้อมูลไบต์คู่และคี่ตามลำดับ เมื่อ พาเล็ทสี ถูกแอดเดรส แอททริบิวต์บิท 1 บิท (สำหรับแต่ละจุดภาพ) จะอยู่ใน พาเล็ทรีจิสเตอร์ ที่แอดเดรสบิท 0 ดังรูป 2.8



รูป 2.8

วีจีเอ 2 สี (VGA Two-color Emulation โหมด 11)

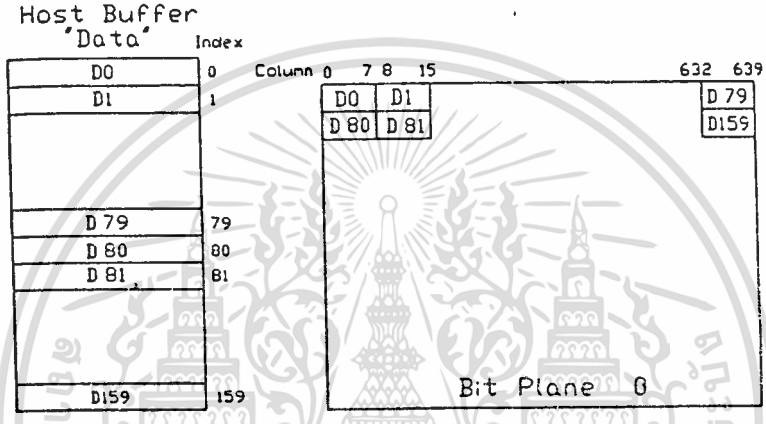
ในโหมด 11 ของ วีจีเอ 1บิทต่อ 1จุดภาพ จะเลือก 1 ใน 2 ของพาเล็ท ค่าที่โหลดลงใน พาเล็ทรีจิสเตอร์ แสดงในตารางที่ 2.10

ตาราง 2.10 ค่าในพาเล็ทที่กำหนดไว้ในโหมด 11 Hex

รีจิสเตอร์	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ค่า	0	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37

ใน หน่วยความจำแสดงผล แต่ละ ไบต์ ประกอบด้วย 8 จุดภาพ หน่วยความจำจะแมป (map) อย่างตรงๆ คือ ใช้เพียง บิทเพลน 0 ในการเก็บข้อมูล เมื่อพาเล็ทสี ถูก แอดเดรส แอททริบิวต์ บิท 1 บิท (สำหรับแต่ละจุดภาพ) จะอยู่ใน พาเล็ท รีจิสเตอร์ ที่ แอดเดรส บิท 0

เป็นการแสดงผลความละเอียด 640*480 จุดภาพ 2สี ดังนั้นจึงต้องการ 1 บิท/จุดภาพ มี แอดเดรส เริ่มต้นที่ A0000 และใช้ หน่วยความจำแสดงผลเพียงบิทเพลน 0 เท่านั้น ดังในรูป 2.9

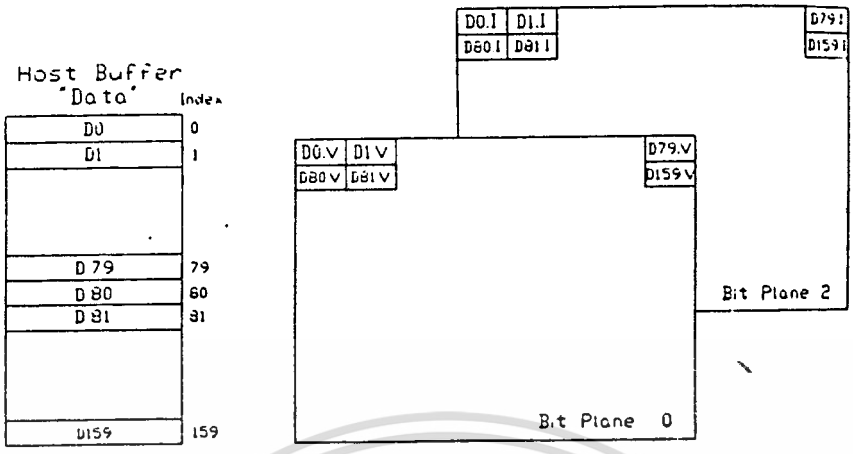


รูป 2.9

โมโนโครมกราฟฟิคส์โหมด (Monochrome Graphics ModelMode F hex)

ใน โหมด F hex มีรหัสขนาด 2 บิท ซึ่งอ้างได้ถึงทุกๆจุดภาพ 2 บิทนี้แทนบิทสัญญาณวิดีโอ และ บิทความเข้มของโมโนโครมแอททริบิวต์ แอททริบิวต์ ขนาด 2 บิท จำนวน 4 แอททริบิวต์ ถูกบรรจุรวมกันเป็น 1 ไบต์ เพื่อโหลดเข้าไปในหน่วยความจำแสดงผล เมื่อ โหมด F hex แอคทีฟ ฮาร์ดแวร์จะโหลด วิดีโอแอททริบิวต์บิตทั้งหมดลงใน บิทเพลน 0 และ แอททริบิวต์บิทที่แสดงความเข้มทั้งหมดลงในบิทเพลน 2

โหมดนี้ จะแสดงผลขนาด 640*350 จุดภาพ 2 สี แต่มีความเข้มด้วย ดังนั้นจึงแสดงความแตกต่างได้ 4 ระดับ ใช้ 2 บิท/จุดภาพ ใช้หน่วยความจำแสดงผลบิทเพลน 0,2 ในการเก็บข้อมูล ดังรูป 2.10



รูป 2.10

โหมด 16 สี (Mode 16 color) โหมด D,E,10H,12H)

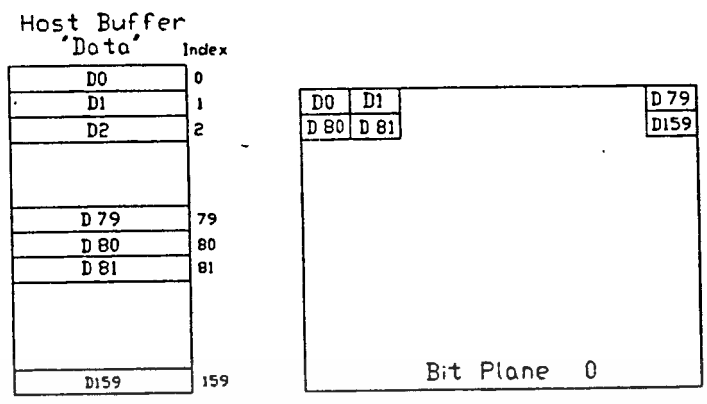
โหมดนี้ประกอบด้วย D,E,10 และ 12 Hex 1จุดภาพ ถูกแทนด้วย 4 บิทใน หน่วยความจำแสดงผล 4บิทนี้อยู่ใน บิทเพลน 0-3 (1 บิท อยู่ใน 1 บิทเพลน) ซึ่งใช้เป็น แอดเดรส ขนาด 4 บิท เพื่อเลือก พาเล็ทรีจิสเตอร์

แต่ละ 2 บิทใน 1 พาเล็ทรีจิสเตอร์ จะแทน สีแดง,เขียว,น้ำเงิน (2บิท/สี) ดังนั้น แต่ละสี จึงมีความเข้มได้ 4 ระดับ ค่าที่เป็นไปได้ของสีเหล่านี้ แสดงในตารางที่ 2.11

ตาราง 2.11 ค่าในพาเล็ทที่กำหนดไว้ในโหมด D,E,10 และ 12 Hex

รีจิสเตอร์	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ค่า	0	1	2	3	4	5	14	7	38	39	3A	3B	3C	3D	3E	3F

โหมดการแสดงผล 16 สีในโหมดนี้ มีความละเอียดต่างกันตามโหมด คือ 320*200 (โหมด D) , 640*200 (โหมด E) , 640*350 (โหมด 10) , 640*480 (โหมด 12) โดยมีลักษณะการเก็บข้อมูลดังรูป 2.11



รูป 2.11

โหมด 256 สี (256-color mode)

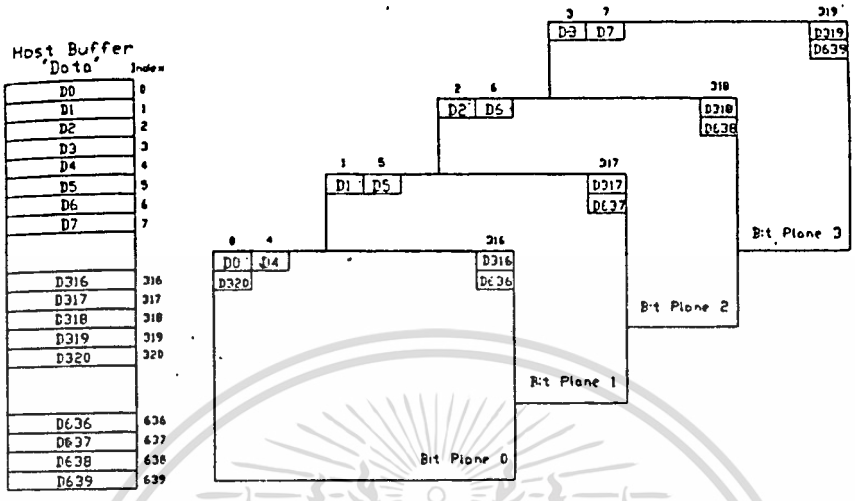
โหมดนี้คือ โหมด 13Hex ซึ่งแสดงได้ถึง 256 สี แต่ละจุดภาพถูกแทนด้วยข้อมูลขนาด 8 บิท ข้อมูลนี้เมื่อถูกโหลดเข้าไปยัง หน่วยความจำแสดงผล จะถูกส่งผ่าน 4 บิท เพลน และจะถูกแยกเป็น 2 ส่วนๆละ 4 บิท ส่วนแรกซึ่งเป็น 4 บิทต่ำ(บิท 0-3)จะ แอดเดรส พาเล็ทรีจิสเตอร์ และ เอาท์พุท 4 บิทต่ำของ พาเล็ทรีจิสเตอร์ ถูกใช้เป็น 4 บิทต่ำ (บิท0-3) สำหรับการแอดเดรสรีจิสเตอร์สี ส่วนที่ 2 คือ บิท 4-7 จาก หน่วยความจำแสดงผล จะถูกใช้เป็น 4 บิทสูง (บิท4-7) ในการแอดเดรสรีจิสเตอร์สี

พาเล็ทรีจิสเตอร์ ถูกจำกัดให้เปลี่ยนแอดเดรส เป็นข้อมูลแบบ หนึ่งต่อหนึ่ง (one-to-one) ดังนั้น ค่าในพาเล็ท จึงเหมือนกับแอดเดรสของมัน ดังแสดงใน ตารางที่ 2.12

ตาราง 2.12 ค่าในพาเล็ทที่กำหนดไว้ในโหมด 13 Hex

รีจิสเตอร์	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ค่า	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

ลักษณะการจัดเก็บข้อมูลของโหมด 13 เป็นดังรูป 2.12



รูป 2.12

บทที่ 3

การหาขอบภาพ

(EDGE DETECTION)

ในระบบการประมวลผลของภาพที่มองเห็นไม่ว่าจะเป็นคนหรือหุ่นยนต์หรือเครื่องจักร ปัญหาสำคัญเบื้องต้นที่เกิดขึ้นในการวิเคราะห์ระบบภาพที่มองเห็น คือการวิเคราะห์หาขอบภาพ (Edge Detection) ขอบของภาพเป็นตัวแสดงถึงขอบของวัตถุ, เส้นขอบเขต ซึ่งมีประโยชน์ในการแบ่งภาพ (Segmentation) และแสดงลักษณะเฉพาะของวัตถุในภาพนั้นๆ

ภาพที่ถูกดิจิไตซ์(Digitized image)จะถูกเก็บในหน่วยความจำเพื่อนำมาประมวลผลต่อ โดยพื้นฐานแล้วภาพของวัตถุสามารถถูกเสนอ หรือเก็บใน 2 รูปแบบ คือ ภาพแบบไบนารี (Binary Image)และ ภาพระดับเทา(Gray scale Image)

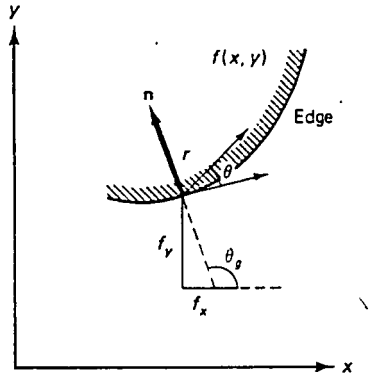
ภาพทางดิจิตอล(Digital Image)จะประกอบไปด้วย ข้อมูลที่เป็นจุดภาพ (Pixel) ความสว่างของแต่ละจุดภาพที่แสดงบนจอภาพจะถูกกำหนดโดยตัวเลข เช่น ในภาพแบบไบนารี่ ความสว่างของภาพ จะใช้ "0" แทนสีดำ และ"1"แทนสีขาว แต่ในภาพระดับเทาความสว่างจะถูกแทนด้วยค่าตัวเลขจำนวนหนึ่ง เช่น ถ้าใช้ 4 บิท สำหรับ 1 จุดภาพ ดังนั้น ค่าระดับเทาที่เป็นไปได้มี 16 ค่าคือ "0" ดำสุด ไปจนถึง "15" ขาวสุด

ภาพทางดิจิตอลนั้นเป็นฟังค์ชั่นทางคณิตศาสตร์ ซึ่งเป็นฟังค์ชั่น 2 ตัวแปร โดยทุกๆ ตำแหน่ง (X,Y) ใดๆในภาพเป็นค่าความเข้มแสง ณ.จุดนั้น ส่วนขอบของภาพเกิดขึ้นที่ขอบเขตระหว่างจุด 2 จุดที่อยู่ติดกันในภาพ ซึ่งมีความแตกต่างของความเข้มของแสงอย่างเห็นได้ชัด ดังนั้นการเกิดขอบของภาพจึงขึ้นอยู่กับการกระจายตัวของค่าความเข้มแสง จุดของขอบภาพสารถถูกพิจารณาได้ว่าเป็นตำแหน่งของจุดภาพที่ระดับความเทา (gray-level) เปลี่ยนอย่างทันทีทันใด หรืออาจให้คำนิยามของ"ขอบภาพ(edge)"ว่าเป็นบริเวณในภาพที่ปรากฏการเปลี่ยนแปลงระดับความเข้มของแสงอย่างชัดเจน เช่น อาจกำหนดให้จุดของภาพในภาพแบบไบนารี่ คือ จุดดำที่มีจุดใกล้เคียงเป็นจุดขาวอย่างน้อย 1 จุด นั่นคือ ตำแหน่งจุดขอบภาพ (m,n) โดยu(m,n)=0 และ g(m,n)=1 ซึ่ง

$$g(m,n) \triangleq [u(m,n) \oplus u(m\pm1,n)].OR.[u(m,n) \oplus u(m,n\pm1)]$$

$\oplus$ คือ กระบวนการเอ็กซ์คลูซีฟออร์ (exclusive-or)

สำหรับภาพที่ต่อเนื่อง f(x,y) อนุพันธ์ของมันจะมากที่สุดในทิศทางของขอบ ดังนั้นเทคนิคอย่างหนึ่งของการหาขอบภาพ คือ วัดเกรเดียนท์ (gradient) ของ f ตาม r ในทิศทาง θ ดังรูป 3.1



รูป 3.1

นั่นคือ

$$(\partial f/\partial r) = (\partial f/\partial x)(\partial x/\partial r) + (\partial f/\partial y)(\partial y/\partial r)$$

$$= f_x\cos\theta + f_y\sin\theta$$

$(\partial f/\partial r)$ สูงสุดเมื่อ $(\partial/\partial\theta)(\partial f/\partial r) = 0$ ดังนั้น

$$-f_x\sin\theta_g + f_y\cos\theta_g = 0$$

$$\theta_g = \tan^{-1}(f_y/f_x)$$

$$(\partial f/\partial r)\max = \sqrt{f_x^2 + f_y^2}$$

ซึ่ง θ_g แสดงถึงทิศทางของขอบภาพ

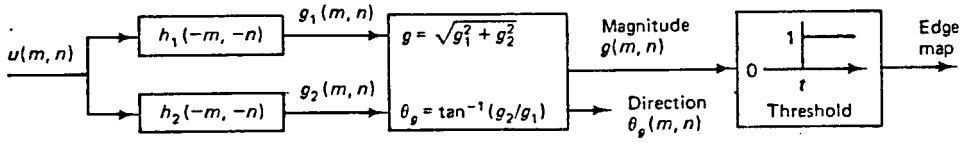
โดยใช้หลักการนี้ ก่อให้เกิดตัวดำเนินการ (operators) ในการหาขอบภาพ 2 ชนิด คือ แบบ เกรเดียนท์ (gradient operators) และแบบ คอมพาส (compass operators) สำหรับภาพดิจิตอลแล้ว ตัวดำเนินการเหล่านี้รวมเรียกว่า ตัวดำเนินการแบบมาสค์ (mask operators)

ให้ H แทน P*P mask สำหรับภาพที่กำหนด U ความสัมพันธ์ระหว่าง U ,H ที่ตำแหน่ง (m,n) เป็นดังนี้

$$\langle U,H\rangle_{m,n} \triangleq \sum_i\sum_j h(i,j)u(i+m,j+n) = u(m,n) \circledast h(-m,-n)$$

3.1 ตัวดำเนินการเกรเดียนท์ (Gradient Operator)

ตัวดำเนินการนี้ถูกแทนโดยคู่มาสค์ H_1 ,H_2 ซึ่งวัดเกรเดียนท์ของภาพ u(m,n) ใน 2 ทิศทางที่ ออโทโกนอล (orthogonal) ดังรูป 3.2



รูป 3.2

โดยที่ $g_1(m,n) \triangleq \langle U, H_1 \rangle_{m,n}$

$g_2(m,n) \triangleq \langle U, H_2 \rangle_{m,n}$

ขนาดและทิศทางของเกรเดียนท์เวคเตอร์ คือ

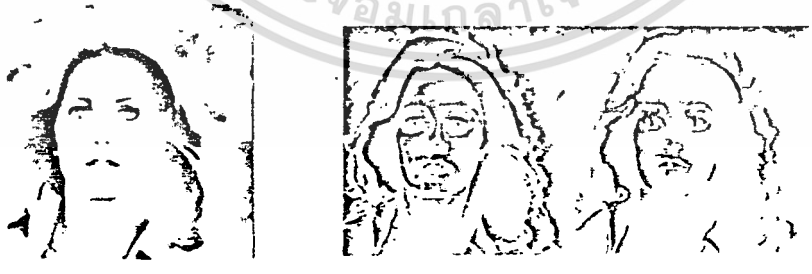
$$g(m,n) = \sqrt{g_1^2(m,n)+g_2^2(m,n)} \triangleq |g_1(m,n)| + |g_2(m,n)|$$

$$\theta_g(m,n) = \tan^{-1}\{ g_2(m,n)/g_1(m,n) \}$$

ตาราง 3.1 แสดงตัวดำเนินการเกรเดียนพรีวิท (Prewitt), โซเบล (Sobel), และไอโซโทรปิค (Isotropic) ซึ่งจะคำนวณความแตกต่างทางแนวราบและแนวดิ่งของผลรวม วิธีการนี้จะช่วยลดสัญญาณรบกวนในข้อมูลได้ จากตาราง จะสังเกตุได้ว่า ตัวดำเนินการเหล่านี้จะให้ผลลัพธ์เป็นศูนย์ สำหรับพื้นที่ ที่เป็นยูนิฟอร์ม (uniform)

	H_1	H_2
Roberts [9]	$\begin{bmatrix} \boxed{0} & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} \boxed{1} & 0 \\ 0 & -1 \end{bmatrix}$
Smoothed (Prewitt [6])	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & \boxed{0} & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & \boxed{0} & 0 \\ 1 & 1 & 1 \end{bmatrix}$
Sobel [7]	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & \boxed{0} & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & \boxed{0} & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Isotropic	$\begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & \boxed{0} & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & \boxed{0} & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$

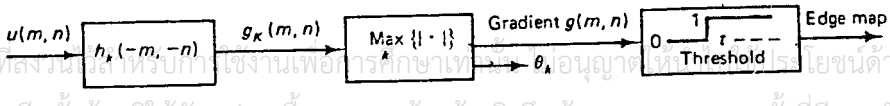
ตาราง 3.1



รูป 3.3 แสดงการหาขอบภาพโดยตัวดำเนินการsobel

3.2 ตัวดำเนินการคอมพาส (Compass Operator)

ตัวดำเนินการนี้จะวัดเกรเดียนท์ในทิศทางที่เลือกไว้จำนวนหนึ่ง ดังรูป 3.4



รูป 3.4

1) $\begin{bmatrix} 1 & 1 & 1 \\ 1 & \boxed{-2} & 1 \\ -1 & -1 & -1 \end{bmatrix}$

2) $\begin{bmatrix} 5 & 5 & 5 \\ -3 & \boxed{0} & -3 \\ -3 & -3 & -3 \end{bmatrix}$ (Kirsch)

3) $\begin{bmatrix} 1 & 1 & 1 \\ 0 & \boxed{0} & 0 \\ -1 & -1 & -1 \end{bmatrix}$

4) $\begin{bmatrix} 1 & 2 & 1 \\ 0 & \boxed{0} & 0 \\ -1 & -2 & -1 \end{bmatrix}$

ตาราง 3.2

ตาราง 3.2 แสดงตัวดำเนินการคอมพาส 4 ชนิด ซึ่งหาขอบไปในทางทิศเหนือ โดยหมุนทวนเข็มนาฬิกาไป 8 ครั้ง ๆ ละ 45 องศา รอบจุดศูนย์กลางของการหมุน ตัวอย่างเช่น ตัวดำเนินการตัวที่ 3 ในตาราง 2.14 จะทำงานดังนี้

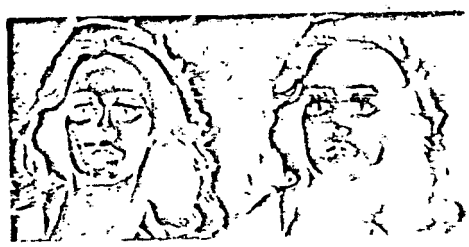
1	1	1	↑		1	1	0	↖		1	0	-1	←		0	-1	-1	↙
0	0	0	(N)		1	0	-1	(NW)		1	0	-1	(W)		1	0	-1	(SW)
-1	-1	-1			0	-1	-1			1	0	-1			1	1	0	
-1	-1	-1	↓		-1	-1	0	↘		-1	0	1	→		0	1	1	↗
0	0	0	(S)		-1	0	1	(SE)		-1	0	1	(E)		-1	0	1	(NE)
1	1	1			0	1	1			-1	0	1			-1	-1	0	

ให้ $g_k(m,n)$ เป็นตัวดำเนินการเกรเดียนท์ ในทิศทาง $\theta_k = \pi/2 + k\pi/4$

$k = 0,1,2,3,4,5,6,7$

เกรเดียนท์ที่ตำแหน่ง (m,n) คือ

$$g(m,n) \triangleq \max_k \{ | g_k(m,n) | \}$$

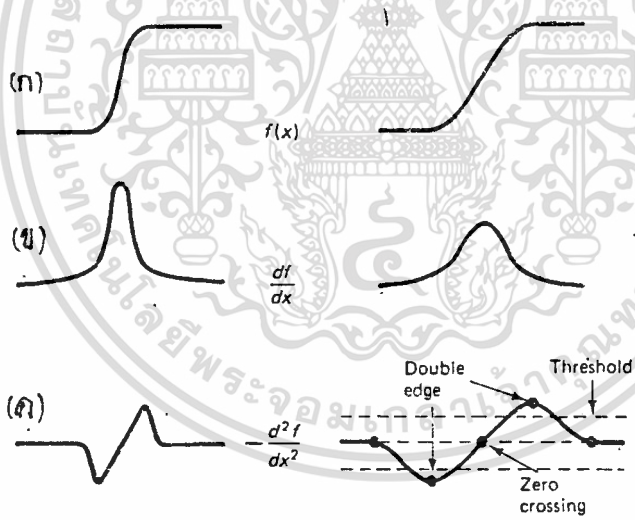


รูป 3.5 แสดงผลของตัวดำเนินการ Kirsch

3.3 ตัวดำเนินการลาปาซและการผ่านจุดศูนย์(Laplace Operator and Zero Crossing)

การประมวลผลของวิธีที่กล่าวมาแล้วนั้นจะให้ผลสูงสุดก็ต่อเมื่อ ความเข้มระดับเทาเปลี่ยนอย่างทันทีทันใด(เหมือนกับสเต็บฟังค์ชั่น(step function)) นั่นคือ เป็นฟังค์ชั่น 2 ตัวแปรที่มีค่าไม่ต่อเนื่อง(discrete) แต่ถ้าพิจารณาฟังชั่นความเข้มแสง เป็นฟังค์ชั่นที่ต่อเนื่อง (continuous) แล้ว วิธีที่กล่าวมาแล้วนั้น จะไม่มีประสิทธิภาพที่ดีพอ ดังนั้นเราจึงใช้วิธีการหาอนุพันธ์อันดับสองมาใช้กับการหาขอบภาพซึ่งให้ผลดีทีเดียว

แนวความคิดของวิธีนี้ก็คือ จะพิจารณาภาพเป็นฟังค์ชั่น 1 มิติ f(x) ที่เกิดจากการตัดขวางภาพ (cross section) แล้ว differentiate f(x) ได้เป็น f'(x) ซึ่งสามารถหาค่ามากที่สุด ณ.ตำแหน่งของขอบภาพ (Edge point) คือบริเวณที่มีการเปลี่ยนแปลงความเข้มของแสงมากที่สุด (slope มีค่ามากที่สุด) ดังรูป 3.6 (ก)และ(ข) และเมื่อ differentiate f'(x) จะได้อนุพันธ์อันดับสองของ f(x) คือ f"(x) จะได้ว่าตำแหน่งที่มีค่าฟังค์ชั่นมากที่สุดในฟังค์ชั่นอนุพันธ์อันดับหนึ่ง f'(x) ก็คือ ตำแหน่งที่มีค่าของฟังค์ชั่นในอนุพันธ์อันดับสองจะมีค่าเป็นศูนย์ เราเรียกจุดนี้ว่า จุดผ่านศูนย์ ดังรูป 3.6 (ค)



รูป 3.6

แต่ในความเป็นจริงแล้ว ภาพที่ได้จากการดิจิไตซ์ จะเป็นฟังค์ชั่นที่ไม่ต่อเนื่องดังนั้นการดิฟเฟอเรนเชียลในทางคณิตศาสตร์จึงใช้ไม่ได้ แต่เราสามารถแก้ไขได้โดยใช้ขบวนการทางคณิตศาสตร์ด้วยวิธี คอนโวลูชั่น (Convolution) และเทคนิคการหาขอบภาพโดยใช้ตัวดำเนินการที่ใช้รูปแบบของฟิลเตอร์ (Filter) สำหรับทำให้ภาพเรียบ (smoothing) ที่อัตราต่างๆ ร่วมกับวิธีการหา อนุพันธ์อันดับสองได้

ตัวดำเนินการแบบลาปาซ $\nabla^2 f$ แทนได้ด้วยสมการ

$$\nabla^2 f = (\partial^2 f/\partial x^2) + (\partial^2 f/\partial y^2)$$

1) $\begin{bmatrix} 0 & -1 & 0 \\ -1 & \boxed{4} & -1 \\ 0 & -1 & 0 \end{bmatrix}$	2) $\begin{bmatrix} -1 & -1 & -1 \\ -1 & \boxed{8} & -1 \\ -1 & -1 & -1 \end{bmatrix}$	3) $\begin{bmatrix} 1 & -2 & 1 \\ -2 & \boxed{4} & -2 \\ 1 & -2 & 1 \end{bmatrix}$

ตาราง 3.3 แสดงการประมาณแบบไม่ต่อเนื่องของลาปาซ

รูป 3.7 แสดงการหาขอบภาพโดยใช้ลาปาซ

เนื่องจากตัวดำเนินการแบบนี้ใช้วิธีหาอนุพันธ์อันดับสอง ดังนั้น จึงเกิดสัญญาณรบกวน (noise) ได้ง่ายกว่าตัวดำเนินการอื่นๆ ข้อเสียอีกอย่างของวิธีนี้คือ $\nabla^2 f$ จะสร้างขอบแบบดับเบิ้ล (double edges) ซึ่งทำให้ไม่สามารถหาทิศทางของขอบภาพได้ จะเห็นได้ว่า วิธีนี้ไม่ใช่วิธีการหาขอบภาพที่ดีนัก

การใช้การหาขอบภาพโดยวิธีจุดผ่านศูนย์ จะมีประสิทธิภาพมากขึ้น เมื่อใช้ร่วมกับเกาซ์เซียนฟังก์ชัน (Gaussian Function) ดังนี้

$$h(m,n) \triangleq c\left[1 - \frac{(m^2 + n^2)}{\sigma^2}\right] \exp\left(-\frac{m^2 + n^2}{2\sigma^2}\right)$$

ซึ่ง σ ควบคุมความกว้างของ Gaussian kernel

c จะนอร์มาไลซ์ (normalizes) ผลรวมของขนาดมาสค์ (mask) ให้เป็นหนึ่ง

h(m,n) คือ ผลตอบสนองต่ออิมพัลซ์ (impulse) ของ แบนด์พาสฟิลเตอร์ (BAND PASS FILTER)

ดังนั้น การหาขอบภาพโดยวิธี "zero-crossing" คือ การใช้วงจรกรองความถี่ต่ำผ่าน (low-pass filter) ที่มีผลตอบสนองเป็น Gaussian Impulse ตามด้วยตัวดำเนินการแบบลาปาซ

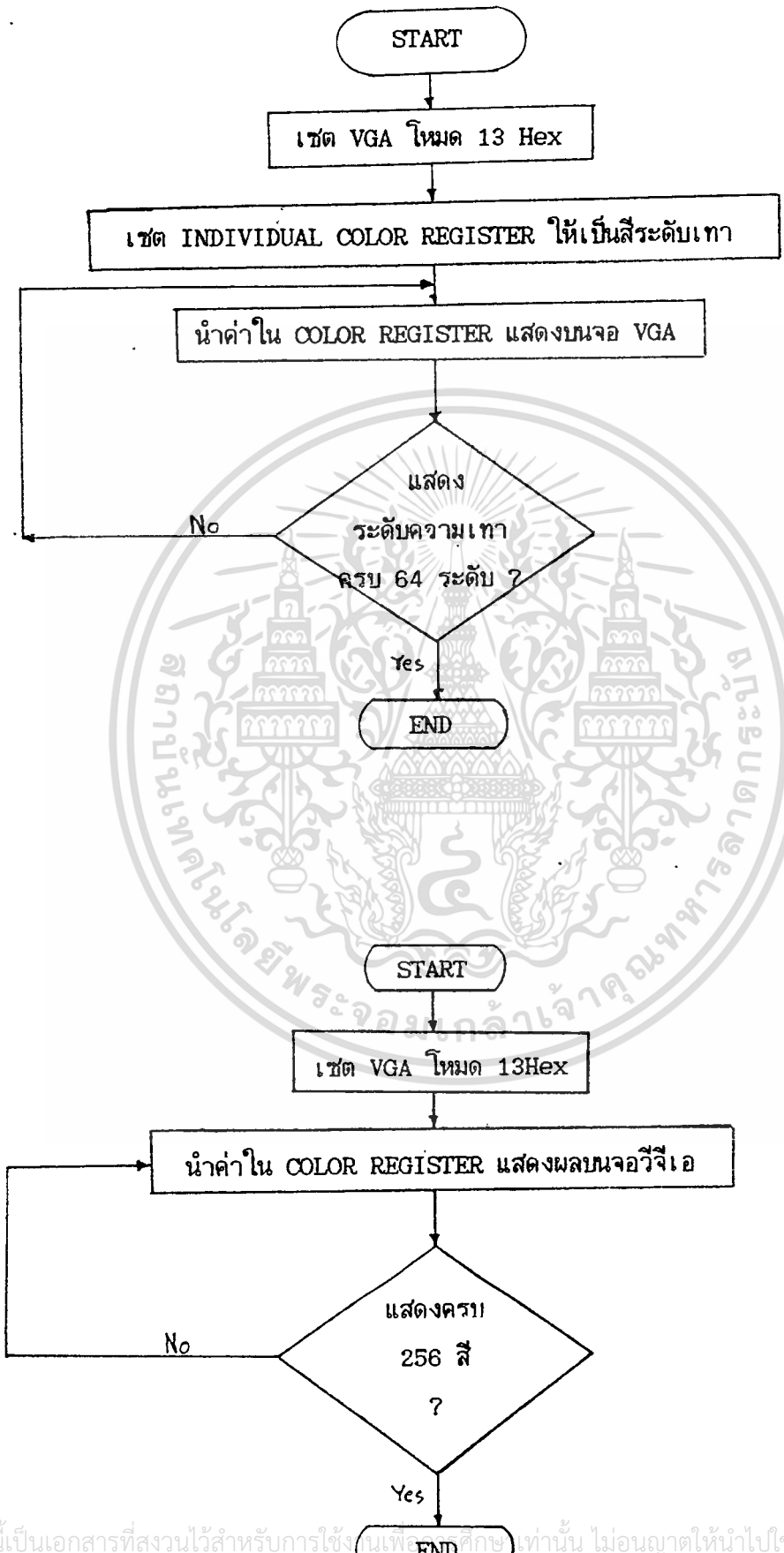
บทที่ 4
การทำงานของโปรแกรมต่างๆ

4.1 PROGRAM DJGRAY.C (DISPLAY GRAY LEVEL)
DJ256.C (DISPLAY 256 COLOR)

เป็นโปรแกรมที่แสดงให้เห็นถึงคุณสมบัติของการ์ดวีจีเอ (VGA CARD) คือ สามารถแสดงสีได้ คราวละ 256 สี (จากทั้งหมด 262,144 สี) และ สามารถแสดงโทน (TONE) สีได้ 64 ระดับเทา (GRAY LEVEL) ทั้งนี้เนื่องมาจากรีจิสเตอร์สี (COLOR REGISTER) ที่ใช้ในการแสดงสีนั้นมี 256 (2^8) ตัว จึงใช้ 8 บิท/จุดภาพ ในการอ้างถึงรีจิสเตอร์สี (ดูในเรื่อง PALLETE REGISTER และ COLOR REGISTER) ในแต่ละ รีจิสเตอร์สี จะมี 18 บิท แบ่งออกเป็น 3 ส่วนๆละ 6 บิท คือแดง,เขียว,น้ำเงิน (แดง 6 บิท,เขียว 6 บิท,น้ำเงิน 6 บิท) ดังนั้นจึงสามารถแสดงสีได้แตกต่างกัน 2^18 = 262,144 สี (256K สี) และแต่ละสีจะมี 2^6=64 ระดับ

ในการที่จะแสดงสีใดๆ ในโหมด 13 HEX นั้น ก่อนอื่นเราต้องโหลดค่าสีที่เราต้องการลงใน รีจิสเตอร์สี เสียก่อน เช่น โปรแกรม "DISPLAY GRAY LEVEL" เราต้องโหลดค่า สีเทาทั้ง 64 สี ลงใน รีจิสเตอร์สี ตัวที่ 0-64 เสียก่อน จากนั้นก็อ้างตำแหน่งของแต่ละรีจิสเตอร์สี แทนการอ้างสี

ฟังค์ชั่นที่ทำการกำหนดค่าลงบน รีจิสเตอร์สี คือ ฟังก์ชัน wrcolor(); ซึ่งอยู่ในโปรแกรม VGATOOL.ASM ที่นำมาแสดงท้ายรายงานนี้



4.2 PROGRAM LOAD.C (DISPLAY IMAGE ON CRT)

เป็นโปรแกรมที่เขียนขึ้นเพื่อดึงไฟล์ข้อมูลที่เป็นรูปภาพ ที่เก็บไว้ในหน่วยความจำ (Disk หรือ Hard disk) ซึ่งมีการจัดเก็บเป็นแบบไบนารี่ (BINARY)ขึ้นมาแสดงผลบนจอมอนิเตอร์ ขนาด 512*256 จุดภาพ ความเข้ม 256 ระดับเทา ที่แอดเดรส D000:0000Hex โดยจะอ่านข้อมูลมาทีละไบต์ จากหน่วยความจำชั่วคราว (Disk หรือ Hard disk) ไปยังหน่วยความจำแสดงผล (จอมอนิเตอร์ที่แอดเดรส D000:0000HEX)โดยโปรแกรมนี้เราใช้ฟังชั่นในภาษาซีคือ fread(); ในการอ่านข้อมูลจากดิสค์ และจะเขียนไปยังจอมอนิเตอร์ โดยใช้คำสั่ง pokeb(segment,offset,value) ซึ่งเป็นการส่งค่า "value" ไปยังหน่วยความจำที่แอดเดรส "segment:offset" ครั้งละหนึ่งไบต์

เรื่องสำคัญที่ควรคำนึงถึงคือ ขนาดของจอมอนิเตอร์ เท่ากับ 512*256 จุดภาพ ซึ่งมีค่า 128 Kไบต์ (131,072ไบต์) แต่เนื่องจาก โครงสร้างทางฮาร์ดแวร์ของวงจรอินเตอร์เฟส (Interface) กับจอภาพ มีการอ้างหน่วยความจำได้เพียง 64 Kbyte ดังนั้น จึงแก้ปัญหาโดยแบ่งหน่วยความจำแสดงผลเป็น 2 เพจ โดยการเซตทางด้านซอฟท์แวร์ โดยใช้ฟังก์ชั่น upJmem(); และ loJmem(); สำหรับเพจบนและเพจล่างตามลำดับ ในการตรวจสอบว่าจะอยู่เพจใดนั้นเราใช้ค่า k ในการตรวจสอบ โดยจะตรวจสอบว่าเกิน 64 Kbyte หรือไม่ ถ้าเกินก็จะสวิช ให้ไปทำงานที่ loJmem();

ฟังก์ชัน cursor();นั้น เป็นฟังก์ชันที่เขียนขึ้นเพื่อกำหนดให้เคอร์เซอร์เลื่อนไปยังแถวและคอลัมน์ที่ต้องการ

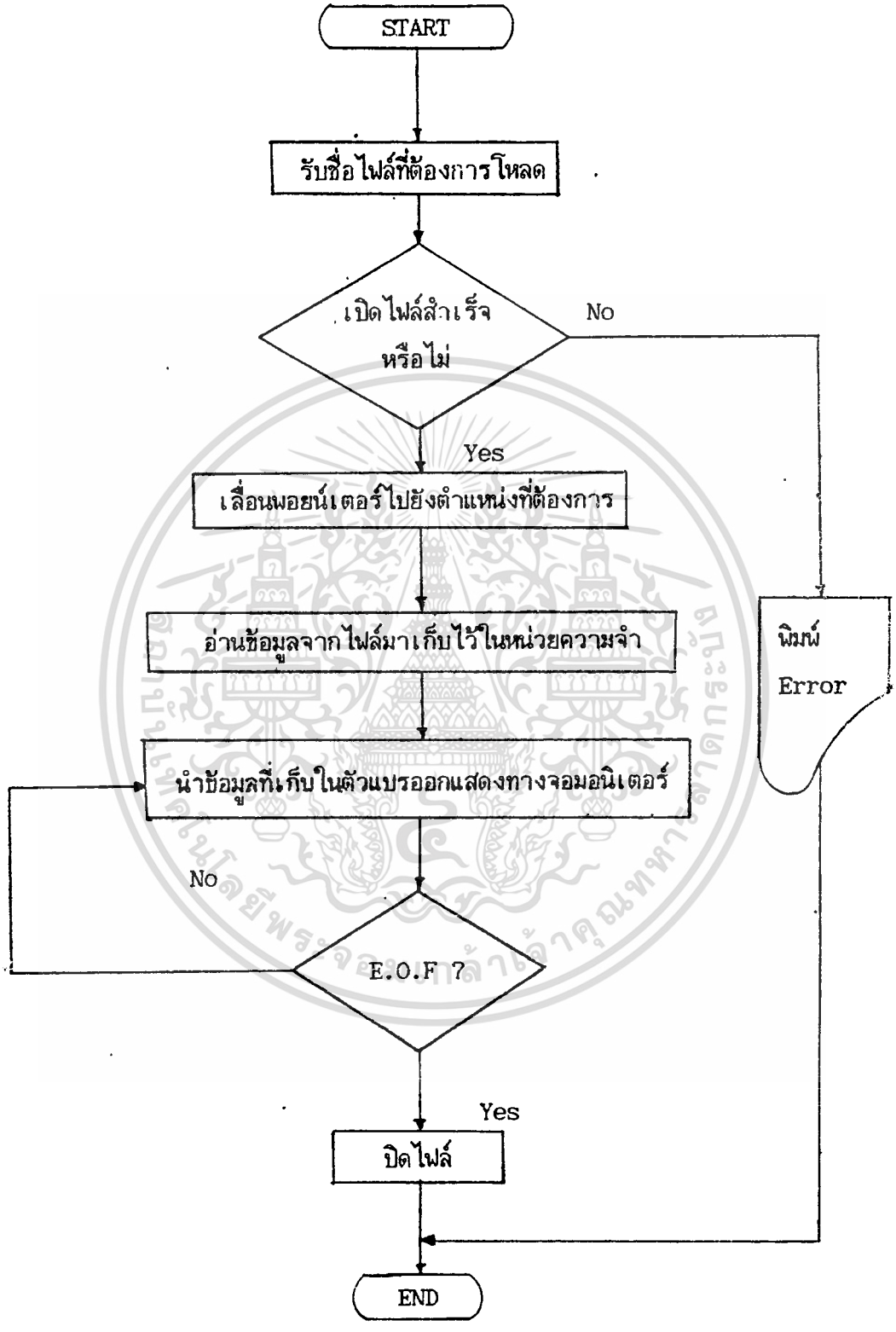
ฟังก์ชัน "upJmem();" หลังจากที่เราเรียกใช้ฟังค์ชั่นนี้แล้ว เมื่อทำการเขียนค่า ลงไปที่ตำแหน่ง เซกเมนต์:ออฟเซต = D000H:0000H ---->D000H:FFFFH แล้ว ค่าที่เขียนลงไป จะไปแสดงอยู่ที่ตำแหน่งบนของจอภาพ

ฟังก์ชัน "loJmem();" หลังจากเรียกใช้ฟังค์ชั่นนี้แล้ว เมื่อทำการเขียนค่าลงไปที่ตำแหน่ง เซกเมนต์:ออฟเซต = D000H:0000H --->D000H:FFFFH แล้วค่าที่เขียนจะไปปรากฏอยู่ที่ ส่วนล่างของจอภาพ

ฟังก์ชัน "farmalloc();" ทำหน้าที่จองเนื้อที่ในหน่วยความจำ

ฟังก์ชัน "weJoff();" ทำหน้าที่บอกให้ฮาร์ดแวร์ของมอนิเตอร์ ยอมให้มีการเขียนค่าลงบนหน่วยความจำแสดงผลของมอนิเตอร์ได้ ซึ่งทำงานตรงกันข้ามกับฟังก์ชัน "weJon();" ในโปรแกรม save.c ซึ่งจะกล่าวต่อไป

สำหรับค่าที่เขียนลงไปบน หน่วยความจำแสดงผล นั้นได้จากการเปิดไฟล์ที่ต้องการโหลด แล้วอ่านมาเก็บไว้ที่หน่วยความจำ ที่ตัวแปร buff[16384] ซึ่งเป็นตัวแปรแบบอะเรย์ (ARRAY) ขนาด 16 KByte ดังนั้นจึงต้องอ่านค่าจากไฟล์แล้วนำไปแสดงคราวละ 16 KByte เป็นจำนวน 8 ครั้ง จึงจะครบ 128 KByte ซึ่งเป็นจำนวนของหน่วยความจำแสดงผลของจอภาพ



4.3 PROGRAM SAVE (SAVE IMAGE TO DISK)

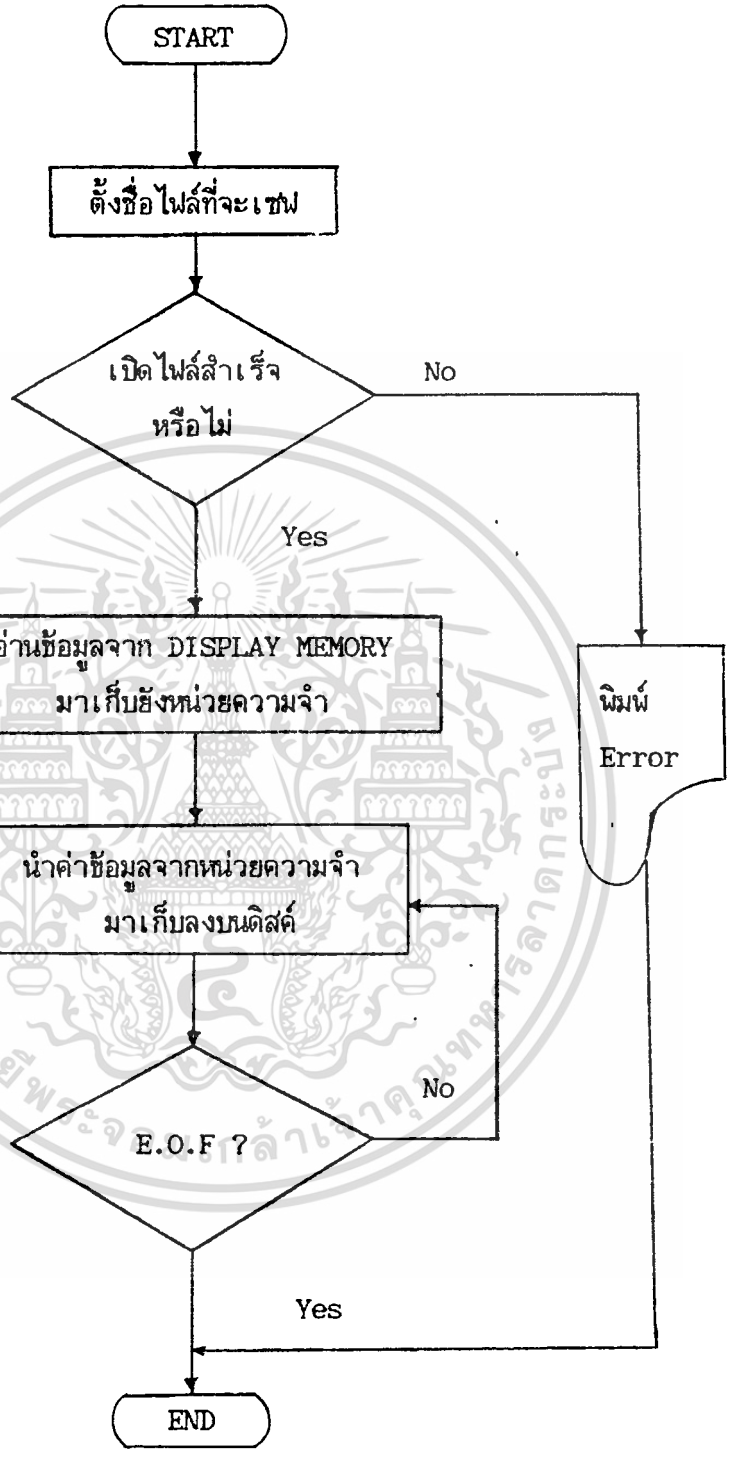
โปรแกรมนี้ ใช้สำหรับบันทึกภาพที่ต้องการ ลงสู่หน่วยความจำชั่วคราว (disk หรือ hard disk) โดยที่ ภาพนั้นต้องเป็นภาพที่กำลังแสดงผลบนจอมอนิเตอร์ ซึ่งภาพที่เกิดขึ้นบนจอนี้ อาจได้มาจากการ โหลดภาพ หรือ จากการถ่ายภาพโดย กล้องดิจิไตซ์ก็ได้ ขบวนการในการดำเนินการของโปรแกรมนี้ ตรงกันข้ามกับ ขบวนการของโปรแกรม LOAD IMAGE ที่กล่าวมาแล้วข้างต้น เพราะ โปรแกรม LOAD IMAGE เป็นการอ่านข้อมูลจากหน่วยความจำชั่วคราว ออกมาแสดงผล แต่โปรแกรม SAVE IMAGE เป็นการเขียนข้อมูล เก็บลงบนหน่วยความจำชั่วคราว

ภาพบนจอมอนิเตอร์ มีความละเอียด 512*256 จุดภาพ ใน 1 จุดภาพ ใช้เนื้อที่ 1 ไบต์ ในการเก็บข้อมูล ดังนั้นภาพ 1 ภาพจะใช้เนื้อที่ในการเก็บข้อมูลถึง 131,072 ไบต์ ในโปรแกรมนี้เราได้จองเนื้อที่ไว้ 16384 ไบต์ ที่ตัวแปร buff[16384] เพื่อใช้ในการถ่ายเทข้อมูลจากหน่วยความจำแสดงผล ลงบนดิสก์ (DISK) ดังนั้น จึงต้องอ่านค่าจากหน่วยความจำแสดงผลแล้วนำไปเก็บลงบนดิสก์ คราวละ 16K ไบต์ เป็นจำนวน 8 ครั้ง เหมือนโปรแกรม LOAD.C

ฟังก์ชัน welon(); เป็นฟังก์ชันที่ทำหน้าที่ บอกให้ฮาร์ดแวร์ของจอมอนิเตอร์ รู้ว่าห้ามให้มีการเขียนข้อมูลใดๆ ลงบนหน่วยความจำแสดงผลในขณะนั้น ฟังก์ชันนี้ใช้เพื่อให้แน่ใจว่าก่อนที่เราจะเซฟภาพที่ต้องการนั้น จะต้องมั่นใจได้ว่า ไม่มีสัญญาณรบกวน หรือ ค่าใดๆ ถูกเขียนลงบนภาพในขณะที่ทำการเซฟ

การใช้ ฟังค์ชั่น farmalloc(); farfree(); upmem(); lomem(); ก็เช่นเดียวกับ โปรแกรม LOAD IMAGE ดังนั้นจะไม่ขอกล่าวซ้ำอีก ส่วนการเขียนข้อมูลลง ดิสค์นั้น ใช้ฟังค์ชั่นในไลบารี่ (LIBRARY) คือ fwrite(buff,16384,1,fp)

โดย buff คือ ตัวแปรที่ใช้ในการถ่ายเทข้อมูล
16384 คือ จำนวนไบต์ที่ต้องการเขียน
1 คือ เขียนครั้งละ 1 ไบต์
fp คือ pointer ที่ใช้ชี้ไฟล์



4.4 PROGRAM DISP (DISPLAY IMAGE ON VGA)

การดิสเพลย์ภาพซึ่งกำลังแสดงอยู่บนจอมอนิเตอร์ ให้ปรากฏบนจอวีจีเอนั้น ใช้หลักการคือ ภาพที่กำลังแสดงอยู่บนจอมอนิเตอร์ นั้นจะมีข้อมูลอยู่บนหน่วยความจำแสดงผลด้วย ที่ตำแหน่ง D000:0000 ถึง D000:FFFF หากเราต้องการให้แสดงบนจอวีจีเอด้วย ก็ทำได้โดยการก๊อปปี้ (COPY) ข้อมูลจาก D000:0000 ถึง D000:FFFF ไปยัง หน่วยความจำแสดงผลของวีจีเอ ซึ่งอยู่ที่ตำแหน่งตั้งแต่ A000:0000 เป็นต้นไป

จากหลักการนี้ เราจะใช้วิธีพอยน์เตอร์ซึ่งมีข้อดี คือ ความเร็ว คือ จะเร็วกว่าการใช้คำสั่ง peekb(); , pokeb(); ซึ่งเป็นฟังก์ชั่นมาก เพราะว่าการเรียกใช้ฟังก์ชั่น จะต้องเสียเวลาในการเซฟค่าแอดเดรสปัจจุบันก่อน จึงจะกระโดด (jump) ไปทำยังฟังก์ชั่นที่ถูกเรียก (call) เมื่อทำเสร็จแล้วก็ต้องรีเทิร์น (return) ค่ากลับซึ่งจะเห็นได้ว่า เสียเวลามากขึ้น

การเรียกใช้พอยน์เตอร์ในภาษาซี (C Language) สามารถทำได้โดยกำหนดดังนี้

```
char far *vgaladdr , *crtladdr ;
```

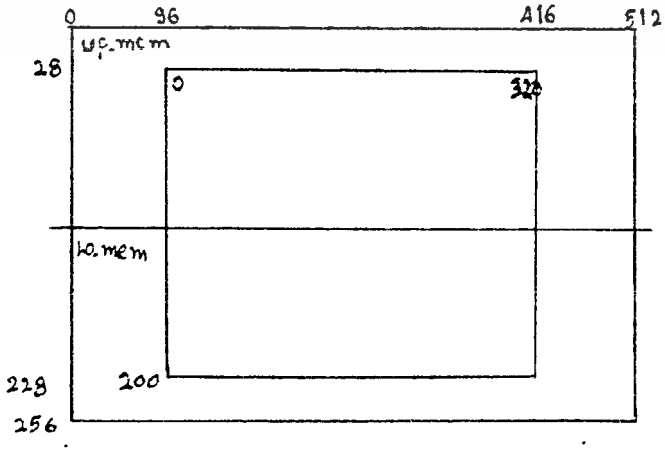
ซึ่งจะหมายถึง การกำหนดให้ vgaladdr และ crtladdr เป็นตัวแปรชนิดพอยน์เตอร์ ขนาด 32 บิต โดยจะชี้ไปยังหน่วยความจำขนาด 1 ไบต์ เช่น

```
vgaladdr = 0xA0000000 ;  crtladdr = 0xD0000000 ;
```

จะเห็นว่า ตัวแปรมีขนาดความยาวถึง 32 บิต และ " *vgaladdr" จะหมายถึงค่าของข้อมูลขนาด 1 ไบต์ ที่แอดเดรส 0xA0000000

ถ้าเขียนว่า *vgaladdr = *crtladdr จะหมายถึง ก๊อปปี้ (copy) ค่าที่แอดเดรส D000:0000 ไปยังแอดเดรส A000:0000 จำนวน 1 ไบต์

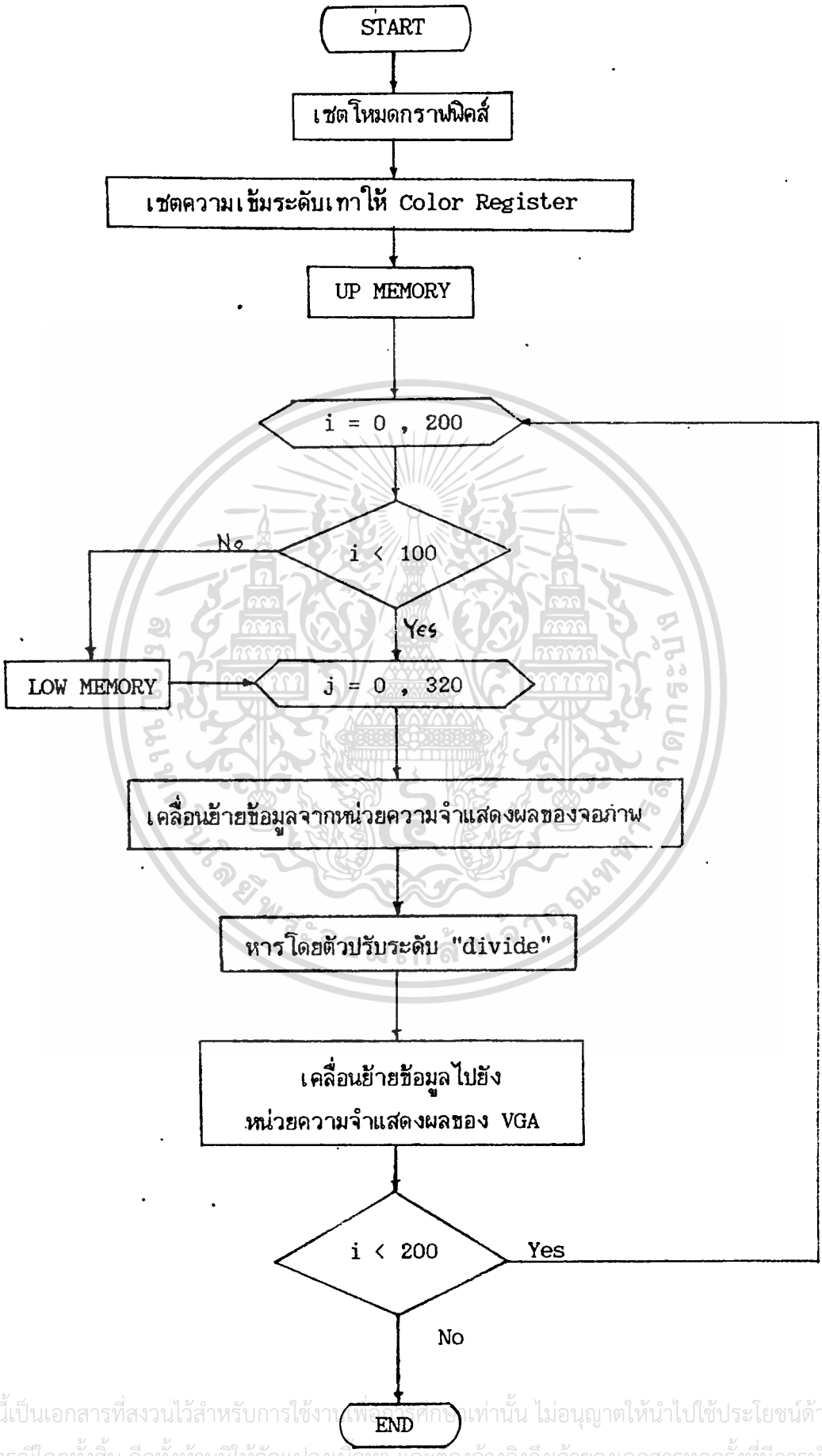
สำหรับค่าของข้อมูลบนจอมอนิเตอร์ ที่ถูกก๊อปปี้ ไปยัง vga นั้นจะถูกหารด้วยตัวแปร " divide " ซึ่งสร้างขึ้นเพื่อเป็นตัวปรับระดับความเข้มของจอวีจีเอ ให้ความเข้มระดับเทามีความกลมกลืนกัน เนื่องจากว่าบนจอมอนิเตอร์มีความเข้มระดับเทาถึง 256 ระดับ แต่บนจอวีจีเอมีความเข้มระดับเทาเพียง 64 ระดับ



แต่เนื่องจากจอมอนิเตอร์ใช้หน่วยความจำมากกว่าจอวีจีเอ (มอนิเตอร์ใช้ 128K ไบต์, วีจีเอใช้ 64K ไบต์) ทั้งนี้เนื่องจากจอมอนิเตอร์มีความละเอียด 512*256 ส่วนจอ วีจีเอ มีความละเอียด 320*200 ดังนั้นในการแสดงผลจึงต้องตัดมาแสดงเพียงบางส่วน โดยเริ่มนำข้อมูลจากจอมอนิเตอร์ ที่ตำแหน่ง แถว 28,คอลัมน์ 96 มาแสดงที่ตำแหน่งแรกของจอ วีจีเอ ดังรูปที่ผ่านมา

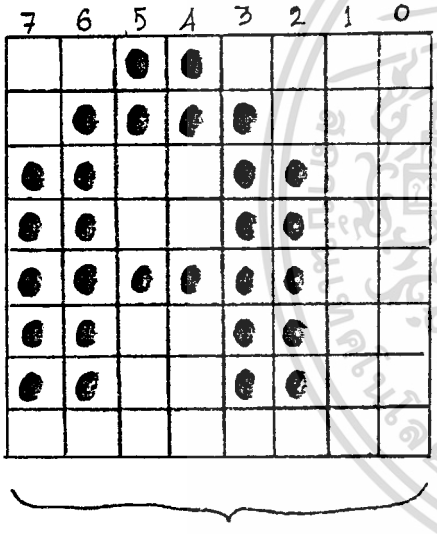
บนจอมอนิเตอร์ มีขนาด 512 * 256 = 128K ไบต์ แต่หน่วยความจำแสดงผลของจอมอนิเตอร์มีขนาดเพียง 64K ไบต์ ดังนั้น จึงต้องแบ่งเป็น 2 เพจ โดยการเซตทางด้านฮาร์ดแวร์ คือ ใช้ฟังก์ชั่น upJmem(); และ loJmem(); ดังที่ได้อธิบายไปแล้วในโปรแกรมที่ผ่านมา

ดังนั้น เราจึงกำหนดให้ crtJaddr เป็นพอยน์เตอร์ ชี้ที่จุดดำ ดังรูปข้างบน ซึ่งมีแอดเดรสอยู่ที่ D000:3860 และ vgaJaddr ชี้ที่ A000:0000 แล้วทำการเคลื่อนย้าย (move) ค่าทีละไบต์จนถึง 320*100 = 32000 ครั้ง จึงทำการเปลี่ยนเพจ โดยการเรียกฟังก์ชั่น loJmem(); แล้วให้ crtJaddr ชี้ที่ D000:0060 แล้วทำการเคลื่อนย้ายข้อมูลต่อ จนถึง 320*200 = 64000 ครั้ง



4.5 PROGRAM WRITECH (WRITE CHARACTER TO IMAGE)

เนื่องจากการเขียนตัวอักษรลงใน โหมดกราฟฟิกส์ แตกต่างจากใน โหมดตัวอักษร ดังที่ได้กล่าวมาแล้ว คือ โหมดกราฟฟิกส์ จะใช้ หน่วยความจำแสดงผล ทั้งหมดในการแสดงผล ดังนันจึงไม่มีที่ว่างเหลือในการเก็บ ฟอนต์ ดังนั้นการที่จะเขียน ตัวอักษรได้ จะต้องจองเนื้อที่หน่วยความจำส่วนอื่น มาใช้ในการเก็บ ฟอนต์ จากนั้นก็นำเอารหัสแอสกี มาหาตำแหน่งที่เก็บรูปแบบ (PATTERN) ของตัวอักษรนั้นๆเพื่อดึงเอาข้อมูลมากำหนด จุดบนจอภาพให้เป็นรูปตัวอักษร เช่น จองหน่วยความจำที่แอดเดรส 1234:0000 แล้วโหลด ฟอนต์ลงบนตำแหน่งนั้น สมมติเราต้องการเขียนตัว "A" ซึ่ง มีรหัสแอสกี 65 ขนาด 8*8 ดังนั้นที่ตำแหน่ง 1234H:(0000H+(65*8)D) = 1234H:0208H จะมี บิทแพทเทิร์นของตัวอักษร "A" อยู่ เพราะฉะนั้นจะได้ 30 78 CC CC FC CC CC 00 มาพล็อตจุดให้เป็นรูปตัว "A" ตามอัลกอริทึม (ALGORITHM) ดังนี้



จากรูปต้องการกำหนดจุดดำของ บิทแพทเทิร์น ของภาพให้ปรากฏบนจอภาพ

แถวบน ข้อมูลคือ 30HEX--> 0011 0000

จะเห็นว่าเราต้องกำหนด จุดดำตรงตำแหน่งที่มี 1

-อ่านค่า 0011 0000 มาเก็บไว้ที่ตัวแปร

-ค่าตัวแปรน้อยกว่า 1000 0000 ไม่ต้องกำหนดจุด

-เลื่อนบิทไปทางซ้าย ตัวแปรจะได้ 0110 0000

-ค่าตัวแปรน้อยกว่า 1000 0000 ไม่ต้องกำหนดจุด

-เลื่อนบิทไปทางซ้ายอีก ตัวแปรจะได้ 1100 0000

-ค่ามากกว่าหรือเท่ากับ 1000 0000 กำหนดจุด

-เลื่อนบิทถัดมาทางซ้ายจนกระทั่งครบ 8 บิท

-อ่านค่าไบท์ถัดมา(78 CC CC FC CC CC 00)

แล้วทำเหมือนข้างต้น

ซึ่งฟังก์ชั่นที่ทำหน้าที่ดังกล่าวนี้ ก็คือ ฟังก์ชั่น plchar(); นั่นเอง

โปรแกรมจะทำการ โหลดฟอนต์ ลงบนหน่วยความจำขนาด 3584 ไบต์ ฟอนต์ขนาด 8*14 นี้นำมาจาก ราชวิถี เวิร์ด เพื่อที่จะให้แสดงผลได้ทั้งภาษาไทยและภาษาอังกฤษโดยใช้คีย์ ESC เป็นตัวสวิช (SWITCH) ในการเลือก จากนั้นก็ทำการรับค่ารหัสแอสกีจากคีย์บอร์ด แล้วส่งค่าไปยัง ฟังค์ชั่น plchar() ทำการหาตำแหน่งของแพทเทิร์นแล้วพล็อตจุดบนจอภาพ นอกจากนี้ยังมีส่วนที่ควบคุมทิศทางของเคอร์เซอร์ (cursor) เช่น การเคลื่อนที่ไปทางซ้าย , เคลื่อนที่ไปทางขวา , เคลื่อนที่ขึ้นบน , เคลื่อนที่ลงล่าง และ ยังมีฟังก์ชั่นคีย์ เช่น แบ็คสเปซ (BACKSPACE) , รีเทิร์น (RETURN) สำหรับควบคุมการเขียนตัวอักษรลงในโหมดกราฟฟิคส์ของวีจีเออีกด้วย

โปรแกรม WRITECH นี้ มีฟังก์ชั่นคีย์ "ESC" เป็นตัวเปลี่ยนโหมดระหว่างภาษาไทยกับภาษาอังกฤษ โดยที่เมื่ออยู่ในโหมดภาษาไทย เมื่อกดคีย์ "ESC" ก็จะเปลี่ยนไปอยู่ในโหมดภาษาอังกฤษแทน และ ถ้าต้องการเปลี่ยนโหมดเป็นภาษาไทยอีก ก็เพียงแค่กดคีย์ "ESC" อีกครั้งเท่านั้น

ข้อควรระวังในส่วนของโปรแกรมช่วงภาษาไทย คือ มีทั้งพยัญชนะ และ สระ ดังนั้นในการเขียนโปรแกรมในโหมดอักษรไทยนั้น เราจึงต้องพิจารณาเป็นสามส่วนคือ

-พยัญชนะ และ สระ ที่อยู่บนบรรทัดเดียวกับพยัญชนะ คือ

```
ช่วง ( chr > 0xa0 ) && ( chr < 0xd7 )
```

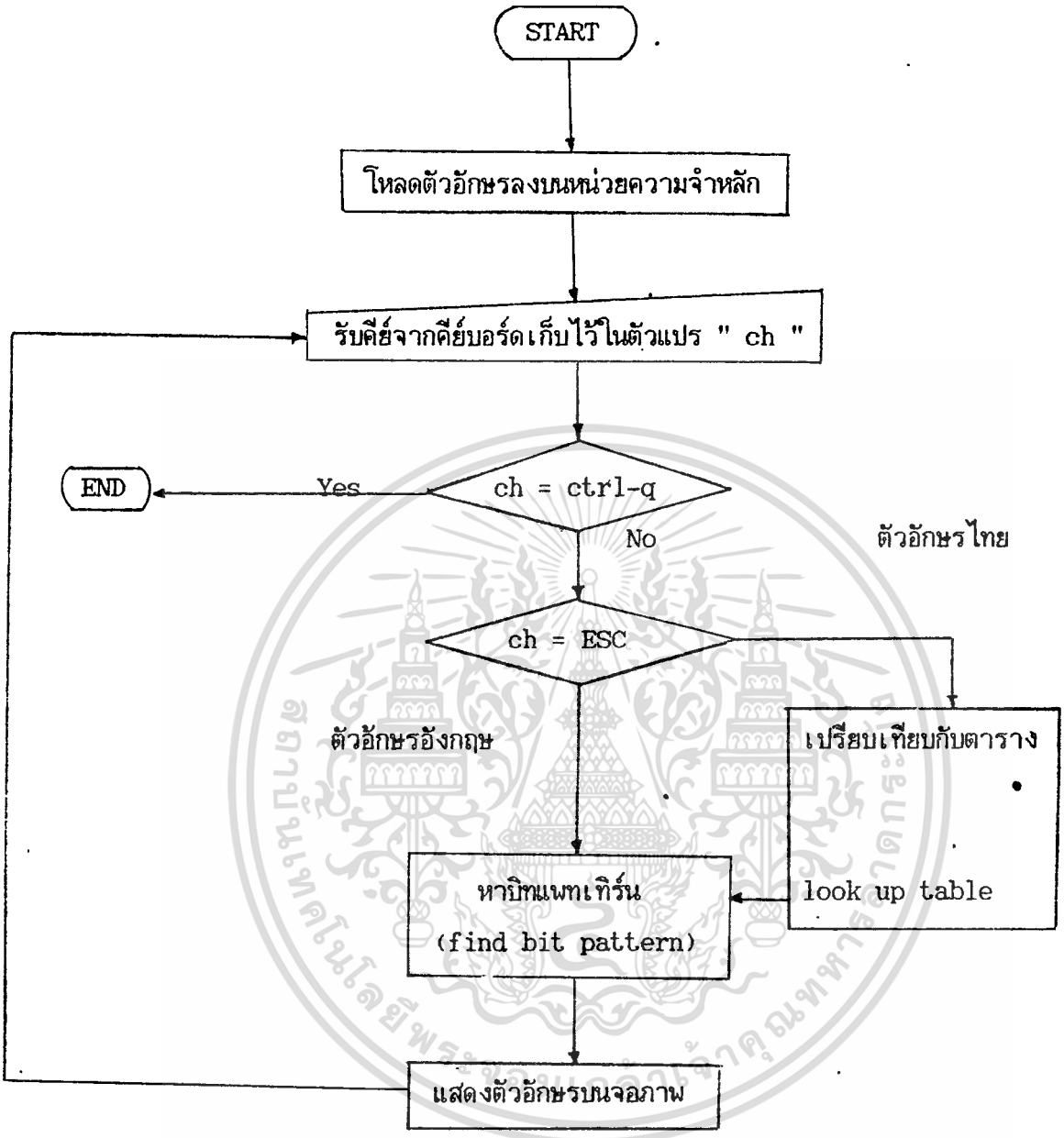
-สระที่อยู่เหนือพยัญชนะ เช่น สระอิ , สระอี คือ

```
ช่วง ( chr > 0xd8 ) && ( chr < 0xff )
```

-สระที่อยู่ใต้พยัญชนะ คือ สระอุ , สระอู คือ

```
( chr == 0xd7 ) || ( chr == 0xd8 )
```

สำหรับฟังก์ชั่นคีย์ต่างๆนั้น เช่น ฟังก์ชั่นคีย์ "BACKSPACE" คือ ch = 0x08 จะทำหน้าที่ลบตัวอักษรที่อยู่หน้าเคอร์เซอร์ 1 ตัว , ฟังก์ชั่นคีย์ "RETURN" คือ ch = \r จะทำหน้าที่รีเทิร์น หรือ ขึ้นบรรทัดใหม่ จะเห็นได้ว่าโปรแกรมนี้มีประโยชน์ คือ สามารถนำไปประยุกต์ เพื่อเขียนตัวอักษร ลงในจอมอนิเตอร์ ซึ่งกำลังแสดงรูปภาพอยู่ เพื่อทำแฟ้มข้อมูลภาพ ซึ่งจะอธิบายในส่วนต่อไป



4.6 PROGRAM EDGE (DETECT EDGE OF IMAGE)

ภาพที่มีความเข้มใกล้เคียงกันในบริเวณเดียวกันทำให้การวิเคราะห์ภาพที่ได้ อาจไม่ตรงกับความเป็นจริงนัก เช่น การวิเคราะห์ภาพถ่ายจากดาวเทียม หรือการพิมพ์ภาพด้วยเลเซอร์พริ้นเตอร์ (LASER PRINTER) มักทำให้ภาพที่ได้ไม่คมชัด ดังนั้นจึงมีการพัฒนาในเรื่องการประมวลผลภาพ (IMAGE PROCESSING) ขึ้น ซึ่งการหาขอบภาพก็เป็นหัวข้อหนึ่งในนั้น

ในโปรแกรมนี้ได้ใช้การหาขอบภาพในการลดจำนวนข้อมูลภาพเพื่อประโยชน์ในการทำแฟ้มข้อมูลภาพ

จากทฤษฎีต่างๆของการหาขอบภาพ (EDGE DETECTION) ในบทที่ 3 เมื่อพิจารณาถึงข้อดี และ ข้อเสียของวิธีต่างๆ แล้ว เราเลือกวิธีการหาขอบภาพ โดยใช้โซเบลโอเปอเรเตอร์ (Sobel Operator) ซึ่งในการพิจารณา จุดภาพ ที่เป็นขอบหรือไม่นั้น จะทำการพิจารณาจาก ความเข้มของจุดภาพใกล้เคียง ทั้งในแนวราบ และ แนวดิ่ง ดังรูป

	jm	j	jp
im	-1	-2	-1
i	0	i,j	0
ip	1	2	1

k1

	jm	j	jp
im	-1	0	1
i	-2	i,j	2
ip	-1	0	1

k2

สมมติว่าเรากำลังพิจารณา จุดภาพ (i,j) ว่าเป็นขอบหรือไม่ เราจะต้องหาจากจุดภาพใกล้เคียงด้วย โดยหาจากผลต่างของความเข้มระหว่างแถวบน,แถวล่าง และ คอลัมน์

แรก,คอลัมน์ หลัง ผลบวกของทั้งสองระนาบ จะเป็นตัวแทนของจุดภาพ (i,j) ซึ่งเป็นขอบภาพ

เราอาจพิจารณาในกรณีง่ายๆ ในการหาความแตกต่างของความเข้มในระนาบเดียว เราสามารถที่จะทราบว่าจุด i เป็นขอบของภาพหรือไม่ โดยการพิจารณาผลต่างของความเข้มที่จุด i+1,i-1 ถ้าหากผลต่างนั้นถึงระดับที่เรากำหนดไว้ ก็จะทำให้จุดนั้นเป็นขอบ จากหลักการดังกล่าวทำให้เขียนโปรแกรมได้เป็น ฟังค์ชั่น edetect();

จากรูป เราแทนค่าในฟังก์ชั่น edetect(); ดังนี้

```
k1 = -ia[im][jm]-2*ia[im][j]-ia[im][jp]
     +ia[ip][jm]+2*ia[ip][j]+ia[ip][jp]
```

และ

```
k2 = -ia[im][jm]-2*ia[i][jm]-ia[ip][jm]
     +ia[im][jp]+2*ia[i][jp]+ia[ip][jp]
```

จากสูตรในบทที่ 3

$$g(m,n) = | g(m,n) | + | g(m,n) |$$

ดังนั้น

$$k = | k1 | + | k2 |$$

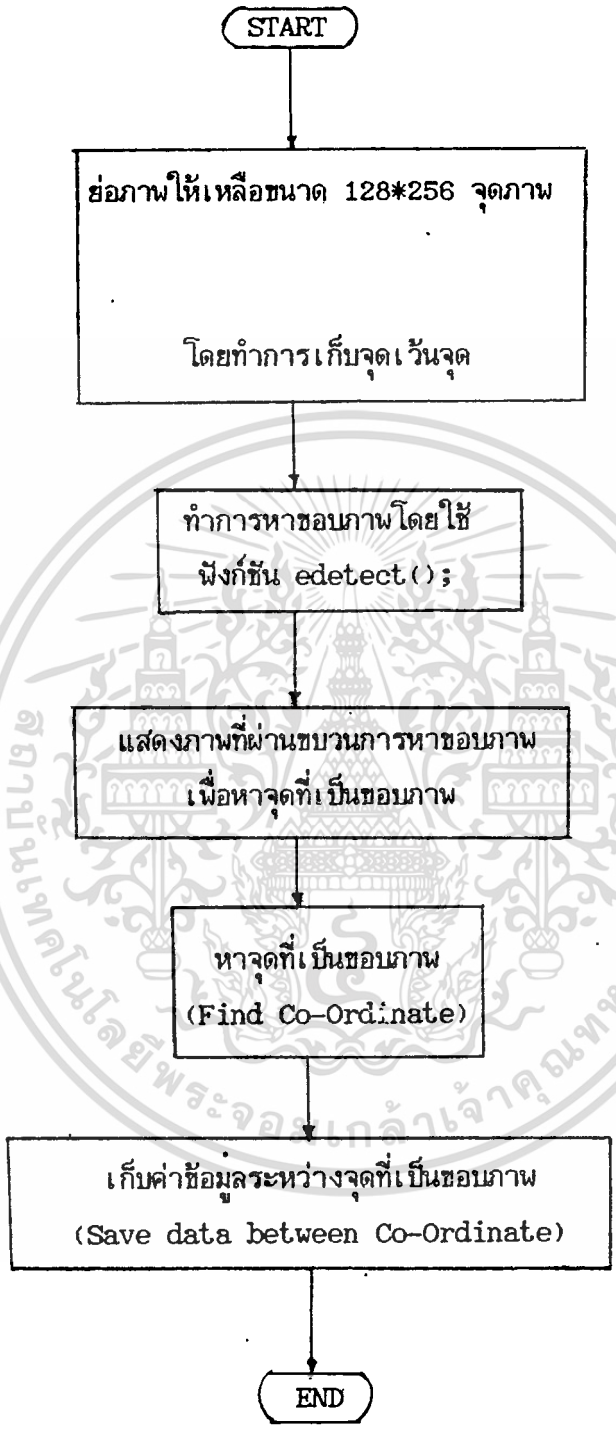
ซึ่งค่า k นี้ ก็คือ ขอบของภาพนั่นเอง

สำหรับการหาขอบภาพของโปรแกรมนี้ ตอนแรกเราต้องการหาขอบภาพกับภาพขนาด 64*128 จุดภาพ แต่เมื่อพิจารณาในเรื่องการเก็บแฟ้มข้อมูลแล้ว จะได้ภาพที่เล็กเกินไป จึงได้เปลี่ยนมาเป็นหาขอบภาพกับภาพขนาด 128*256 จุดภาพ ซึ่งก็ได้ภาพที่มีขนาดเป็นที่น่าพอใจ นอกจากนี้ ยังสามารถลดขนาดของภาพได้ถึง 40-60 % ทำให้ลดพื้นที่ในการเก็บข้อมูลไปได้มากทีเดียว

<u>หลักการทำงาน</u>

โปรแกรมนี้จะทำการหาขอบภาพ กับภาพที่กำลังแสดงอยู่บนจอมอนิเตอร์ในขณะนั้น โดยจะทำการเก็บค่าข้อมูลบนจอมอนิเตอร์ มาเก็บไว้ในตัวแปร "ia" โดยเก็บจุดเว้นจุด เพราะเราต้องการภาพขนาด 128*256 จุดภาพ จากจอภาพขนาด 256*512 จุดภาพ จากนั้นก็จะส่งข้อมูลในตัวแปร "ia" ไปทำการหาขอบภาพ โดยใช้ฟังก์ชัน edetect(); ซึ่งการทำงานของฟังก์ชั่นนี้ ได้อธิบายอย่างชัดเจนแล้วในตอนต้น จึงไม่ขอกล่าซ้ำอีก

เมื่อผ่านขบวนการหาขอบภาพเรียบร้อยแล้ว ขั้นตอนต่อไป คือ แสดงภาพที่ผ่านการหาขอบภาพ เพื่อทำการหาจุดโคออร์ดิเนต (Co-Ordinate) ของขอบภาพ โดยจะทำการเก็บค่าข้อมูลเป็นแถวๆ ซึ่งจะทำการเก็บข้อมูลในแต่ละแถว ดังนี้ ในแต่ละแถว จะมีจุดภาพ ซึ่งเป็นขอบของภาพอยู่ 2 จุด ซึ่งโปรแกรมที่เขียนขึ้น จะตรวจหา (detect) ทั้ง 2 จุดนี้ แล้วนำมาเก็บไว้ โดยจะเก็บจุดซึ่งเป็นขอบแรกก่อน แล้วจึงเก็บจุดที่เป็นขอบหลัง หลังจากนั้น จึงเก็บข้อมูลระหว่างทั้ง 2 จุดนี้ จากนั้นจะเก็บข้อมูลในแถวต่อไป จนครบทั้ง 128 แถว แต่ก่อนที่เราจะเซฟข้อมูลเหล่านี้ เพื่อให้แน่ใจว่า ข้อมูลที่ต้องการถูกต้องครบถ้วน โปรแกรมส่วนต่อไป จะแสดงภาพที่ผ่านการหาขอบภาพมาอย่างสมบูรณ์ นั่นคือ จะมีเพียงเฉพาะรูปภาพเท่านั้น (background ตัดทิ้งหมด) เมื่อภาพที่แสดงถูกต้องครบถ้วน เราก็จะทำการเซฟข้อมูลลงในไฟล์ที่เราเป็นผู้กำหนดชื่อตามต้องการ



4.7 PROGRAM POP2.C (POP UP IMAGE FROM EDGE DETECTION)

ในการทำดาต้าเบส (Database) เกี่ยวกับภาพ ถ้าหากเราจะเขียนตัวอักษรทับลงในภาพ แล้วทำการเซฟทั้งภาพและข้อมูล จะทำให้สิ้นเปลืองเนื้อที่ และเสียเวลาในการโหลดภาพนาน ในทางปฏิบัติแล้ว จะเก็บภาพและ text file แยกกัน และสามารถโหลดขึ้นมารวมกันได้ โดยข้อมูลใน text file จะมีความสัมพันธ์กับภาพ เช่น อาจเป็นที่อยู่ หรือข้อมูลอื่นๆ ซึ่งมีประโยชน์ในการทำแฟ้มข้อมูลบุคคล

โปรแกรมนี้ เป็นโปรแกรมที่ทำหน้าที่ดึงภาพที่ผ่าน ขบวนการ " การหาขอบภาพ " (EDGE-DETECTION) มาแสดงผลบนจอมอนิเตอร์ที่มีความละเอียด 512*256 จุดภาพ แต่ภาพที่แสดงผลนั้นเป็นภาพที่มาจากการหาขอบภาพโดยวิธีของ โซเบล(SOBEL) ได้ภาพที่มีความละเอียด 256*128 จุดภาพ เหตุที่เราแสดงผลเพียงเศษหนึ่งส่วนสี่ของจอภาพ ก็เพราะว่าในส่วนที่เหลือของจอภาพนั้น เราดึงเอาข้อความจากดาต้าเบสไฟล์(DATABASE FILE) ซึ่งเราเก็บไว้ใน ดีเบส(DBASE) มาแสดงผล โดยสรุปแล้วโปรแกรมนี้ทำหน้าที่เป็นแฟ้มข้อมูลภาพนั่นเอง คือ เมื่อเราเรียกโปรแกรมนี้ขึ้นมาเครื่องจะให้เราใส่ชื่อไฟล์ของบุคคลที่ต้องการค้นหา ไฟล์นี้ก็คือ ภาพที่ผ่านขบวนการ "การหาขอบภาพ" ซึ่งมีเอ็กซ์เทนชั่น(EXTENTION) เป็น .img เมื่อเราใส่ชื่อไฟล์เข้าไปแล้ว เครื่องก็จะโหลดเอารูปนั้นเข้ามาแสดงผลบนจอภาพ หลังจากนั้นเครื่องจะโหลดข้อความที่เป็นรายละเอียดเกี่ยวกับรูปภาพนั้นขึ้นมาทันที ซึ่งข้อความที่แสดงนั้นเรียกจากข้อมูลในดีเบส เช่น ชื่อ,ที่อยู่, เลขประจำตัว, เบอร์โทรศัพย์ เป็นต้น

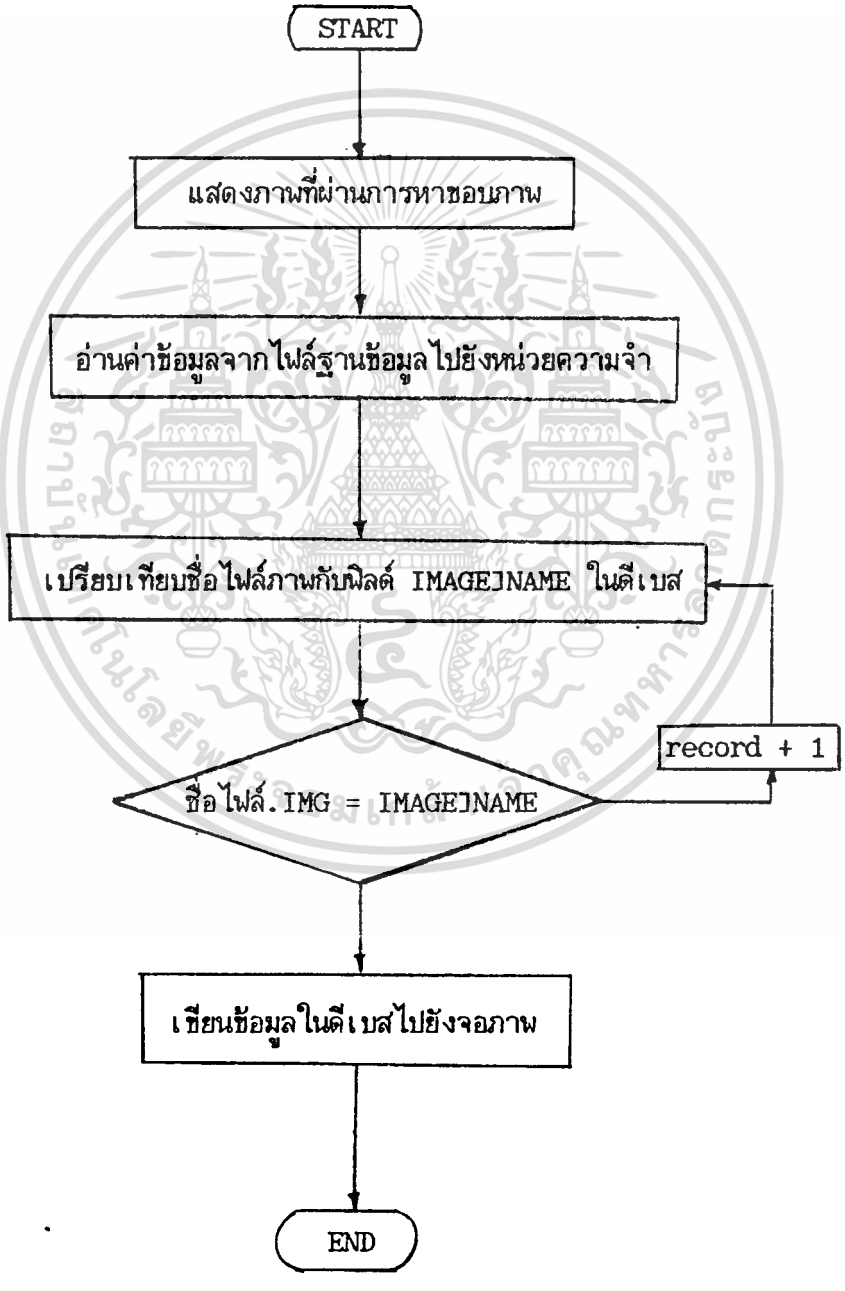
หลักการทำงาน

เมื่อเราใส่ชื่อไฟล์ที่ผ่านการ " การหาขอบภาพ " แล้ว โปรแกรมก็จะไปเปิดไฟล์แล้วนำมาเก็บในตัวแปรซึ่งเราได้จองเนื้อที่ไว้แล้ว หลังจากนั้นเครื่องจะโหลดแบ็คกราวด์ขึ้นมา (เป็นสีซึ่งเรากำหนดโดยค่าแอททริบิวต์) ต่อมาก็จะโหลดเอาภาพที่ต้องการขึ้นมาแสดง จากโปรแกรม จะเห็นว่าในไฟล์ที่ผ่านขบวนการ "การหาขอบภาพ" มาแล้วนั้น เราจะทำการเก็บเป็นแถว คือ ในแถวแรกจะเก็บขอบของภาพทั้ง 2 ด้านติดต่อกันไป แล้วจึงเก็บข้อมูลระหว่างทั้ง 2 จุดนั้น เมื่อข้อมูลในแถวหมดแล้วจึงขึ้นแถวใหม่ แล้วเก็บขอบภาพทั้ง 2 ด้านก่อนแล้วจึงเก็บข้อมูลระหว่าง 2 จุดนั้น เป็นเช่นนี้เรื่อยไปจนครบ 128 แถว ดังนั้นโปรแกรมนี้ จึงต้องเรียกจุดแรกมาเก็บ ไว้ในตัวแปร "SOURCE" เพื่อเป็นพอยน์เตอร์ ตามมาด้วยจุดหลังในแถวเดียวกัน เก็บไว้ในตัวแปรชื่อ "DEST" แล้วจากนั้น จึงให้ข้อมูลระหว่างทั้ง 2 จุดนั้น แสดงบนจอภาพ เป็นเช่นนี้ทุกๆแถว เราจะได้ภาพที่สมบูรณ์ขึ้นมาแสดง

เมื่อโหลดภาพเรียบร้อยแล้ว โปรแกรมก็จะไปเปิดไฟล์ฐานข้อมูลโดยอัตโนมัติ แล้วดึงเอาข้อมูลที่ตรงกับชื่อไฟล์ภาพนั้นมาแสดง วิธีการตรวจเช็คว่าเป็นไฟล์ที่ต้องการหรือไม่ ก็คือ ข้อมูลที่เราเก็บไว้ในดีเบส จะต้องมีฟิลด์หนึ่งเป็นชื่อ ไฟล์ภาพ.img (ซึ่งเป็นตัวเชื่อมโยงกับภาพนั่นเอง) การเปรียบเทียบชื่อไฟล์ภาพกับชื่อไฟล์ในดีเบสเราใช้ฟังชั่น STRNCMP(); ทำหน้าที่นี้ เมื่อชื่อไฟล์ภาพตรงกับชื่อในฐานข้อมูล โปรแกรมก็จะดึงเอาข้อมูลนั้นมาแสดง แต่ถ้าไม่มีชื่อนั้น

โปรแกรมก็จะฟ้องออกมาว่า ไม่มี ให้ผู้ใช้เรียกใหม่ สำหรับโปรแกรมส่วนที่แสดงข้อมูลในดีเบส ให้ปรากฏบนจอมอนิเตอร์นั้น เราใช้ฟังก์ชั่นpJch(); ในการนำข้อมูลไปแสดงผลบนจอมอนิเตอร์ ซึ่งฟังก์ชั่นนี้ได้อธิบายไว้แล้วในโปรแกรม WRITECH:C ดังนั้นจึงไม่ขอกล่าวในส่วนนี้อีก โปรแกรมส่วนที่เหลือ จึงเป็นการจัดรูปแบบการแสดงข้อมูลบนจอให้สวยงาม

สำหรับ ฟังก์ชั่น weJon(); weJoff(); upJmem(); loJmem(); cursor() ; pJch() ; ได้อธิบายไปแล้วในบทที่ผ่านมา จึงไม่ขอกล่าวซ้ำในที่นี้อีก



บทที่ 5

บทสรุปและวิจารณ์

วิทยานิพนธ์เรื่อง " คอมพิวเตอร์กราฟฟิก " นี้ แบ่งเนื้อหาออกเป็น 2 ส่วนใหญ่ๆ คือ ในเรื่องเกี่ยวกับอีจีเอ/วีจีเอ และ ในเรื่องเกี่ยวกับการประมวลผลข้อมูลภาพ ด้วยวิธี " การหาขอบภาพ " (Edge-Detection) ดังนั้น จะสรุปได้เป็น 2 ส่วน ดังนี้

1. ในโหมด 13Hex ซึ่งเป็นโหมดกราฟฟิกของวีจีเอ สามารถแสดงสีได้พร้อมกันถึง 256 สี จากสีที่ให้เลือกทั้งหมด 256K สี (262,144 สี) และ สามารถแสดงระดับความเทาได้ถึง 64 ระดับ ซึ่งจากความสามารถของวีจีเอดังที่กล่าวมานี้ ทำให้ภาพ (image) ที่แสดงบนวีจีเอ มีความต่อเนื่องกลมกลืนกัน ประกอบกับสัญญาณที่ถูกส่งไปยังจอวีจีเอนั้น ได้มาจาก ดิจิตอลทูอนาล็อกคอนเวิร์เตอร์ (digital-to-analog convertor) ดังนั้น ภาพที่แสดงบนจอวีจีเอนี้จึงเป็นภาพที่เหมือนจริง ซึ่งมีประโยชน์ในการวิเคราะห์ หรือ ประมวลผลภาพบนจอวีจีเอนี้

2. การประมวลผลภาพด้วยวิธี "Edge-Detection" โดยตัวดำเนินการโซเบล (Sobel) นี้ ภาพที่ได้จะมีเส้นขอบภาพที่มีลักษณะสมบูรณ์ เห็นได้ชัดเจน ไม่มีส่วนที่ขาดตอน และเวลาที่ใช้ในการวิเคราะห์หาขอบภาพก็ไม่มากนัก และเมื่อนำภาพที่ได้จากการหาขอบภาพมาใช้ประโยชน์ในการทำแฟ้มข้อมูลภาพ โดยทำการเก็บข้อมูลของภาพเฉพาะภาพที่ต้องการเท่านั้น ก็จะทำให้สามารถลดเนื้อที่ในการเก็บข้อมูลของภาพ 1 ภาพได้ถึง 40 - 60 % จากภาพต้นแบบ แต่ข้อเสียของวิธีนี้ก็คือ เส้นขอบภาพยังมีความหนาและมีสัญญาณรบกวนอยู่ และภาพที่เราใช้ในการทำแฟ้มข้อมูลประวัติบุคคล ก็ต้องเป็นภาพถ่ายที่มีสีพื้นของภาพ (background) แตกต่างจากรูปถ่ายของคน (foreground) อย่างเห็นได้ชัด มิฉะนั้นแล้วจะทำให้ไม่สามารถหาขอบภาพที่แท้จริงได้

ภาคผนวก

PROGRAM DISPLAY 64 GRAY SCALE

```
#include<dos.h>
dlgray()
{  int i,j,k;
   char far *vgaladdr = 0xA0000000;
   extern void far wrcolor();
        lAX = 0x0013;
        geninterrupt(0x10);
        for(i=0;i<64;i++)
        wrcolor(i,i,i,i);
        for(i=0;i<200;i++)
        for(j=0;j<64;j++)
        for(k=0;k<5;k++)
          {   *vgaladdr = j;
              vgaladdr++; }
}
```

PROGRAM DISPLAY 256 COLOR

```
#include<dos.h>
dl256()
{  int i,j,k,l;
   char far *vgaladdr = 0xA0000000;
        lAH = 0x00;lAL = 0x13;
        geninterrupt(0x10);
        for(i=0;i<4;i++)
        for(j=0;j<50;j++)
        for(k=0;k<64;k++)
        for(l=0;l<5;l++)
         {   *vgaladdr = (i*64)+k;
             vgaladdr++; }
}
```

PROGRAM LOAD IMAGE

```
#include<stdio.h>
#include<dos.h>
#include<alloc.h>

load()
{
   char far *ptralloc;
   char far *buff;
   int i,j;
   char name[30];
   long int k = 0L;
   FILE *fp;
   extern void far up]mem();
   extern void far lo]mem();
   extern void far cursor();

        cursor(20,40);
        printf("%c%c%c%c%c%c%c%c%c%c%cFrom  file%c%c%c%c%c%c%c%c%c%c%c",
        214,196,196,196,196,196,196,196,196,196,196,196,196,196,
        196,196,196,196,196,196,196,196,183);
        cursor(21,40);
        printf("%c",186);
        for(i=0;i<30;i++)
            printf("%c",32);
        printf("%c",186);
        cursor(22,40);
        printf("%c",211);
        for(i=0;i<30;i++)
            printf("%c",196);
        printf("%c",189);
```

```
    welcoff();
    if((buff = farmalloc(16384)) == NULL)
    {   cursor(23,40);
        printf("error allocate");
        exit(0);
    }
    ptralloc = buff;
    cursor(21,43);
    gets(name);
    if ((fp = fopen(name,"rb")) != NULL)
    {   upimem();
        for (i=0;i<8;i++)
        {   fseek(fp,(long)i*16384,0);
            fread(buff,16384,1,fp);
            for (j=0;j<16384;j++)
            {   pokeb(0xd000,(long)j+k,*buff);
                buff++;
            }
            buff = ptralloc;
            k = k+16384L;
            if (k>65535L)
            {  loimem();
               k=0L;
            }
        }
        fclose(fp);
        farfree(ptralloc);
        cursor(23,35);
        printf("Load complete]]F-10 for return to Menu  ");
    }
   else { cursor(23,35);
          printf("Error]]ENTER for Retry:CTRL-J for Quit  ");}
}
```

PROGRAM SAVE IMAGE

```
#include<stdio.h>
#include<alloc.h>

save()
{
   int i,j ;
   long int  k=0L ;
   FILE *fp ;
   char name[15];
   char far *buff,*ptralloc;
   extern void far welon  ();
   extern void far weloff ();
   extern void far uplmem ();
   extern void far lolmem ();
   extern void far cursor();

        cursor(20,40);
        printf("%c%c%c%c%c%c%c%c%c%c%c%cSave  To%c%c%c%c%c%c%c%c%c%c%c%c",
        214,196,196,196,196,196,196,196,196,196,196,196,196,196,
        196,196,196,196,196,196,196,196,196,196,183);
        cursor(21,40);
        printf("%c",186);
        for(i=0;i<30;i++)
            printf("%c",32);
        printf("%c",186);
        cursor(22,40);
        printf("%c",211);
        for(i=0;i<30;i++)
            printf("%c",196);
        printf("%c",189);
        if(( buff = farmalloc(16384)) == NULL)
```

```
{   cursor(23,40);
    printf(" error allocate");
    exit(0);
}
ptralloc = buff;
cursor(21,43);
gets (name) ;
welon ();
if ((fp = fopen(name,"wb")) != NULL)
{   uplmem();
    for (i = 0; i<8 ; i++)
    {   for (j=0; j<16384; j++)
        {
          *buff = peekb(0xd000,(long) j+k) ;
          buff++;
        }
        buff = ptralloc;
        fseek(fp,(long) i*16384,0) ;
        fwrite(buff,16384,1,fp) ;
        k = k + 16384L ;
        if (k>=65535L)
        {  lolmem ();
           k=0L;
        }
    }
    fclose (fp) ; farfree(ptralloc);
    cursor(23,40);
    printf("Save complete]] F-10 for return to Menu   ");
}
else
{   cursor(23,40);
    printf("Error]]ENTER for Retry:CTRL-J for Quit   ");
}
```

```
}
```

PROGRAM WRITE CHARACTER TO GRAPHIC MODE

```
#include<stdio.h>
#include<dos.h>
#include<conio.h>

#define attrib 0x50
#define TRUE  1
#define FALSE 0
#define ESC   0x1B

char prev;
char buff1[3584];


wrlchar()
{
   int i,j,value,enter;
   int thmode = FALSE;
   FILE *fp;
   long k;
   long offset=(8*320); char ch,chr;
   char far *vgaladdr,*crtladdr;
   char thaitable[] = {
        0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,
        0x08,0x09,0x0a,0x0b,0x0c,0x0d,0x0e,0x0f,
        0x10,0x11,0x12,0x13,0x14,0x15,0xcb,0x17,
        0x1b,0x19,0x1a,0xae,0x1c,0x1d,0x1e,0x1f,
        0x20,0x23,0x2e,0x32,0x33,0x34,0xeb,0xa5,
        0x36,0x37,0x35,0x39,0xbf,0xa2,0xd3,0xba,
        0xa6,0x2a,0x2f,0x5f,0xbe,0xb4,0xd7,0xdb,
```

```
        0xa3,0xb3,0xa9,0xc4,0xb0,0xa8,0xc9,0x24,
        0x31,0xc2,0x2e,0xa7,0xad,0xac,0xd2,0xaa,
        0xdf,0xb1,0xe3,0xc6,0xc5,0x3f,0xe4,0xd6,
        0xab,0x30,0xaf,0xa4,0xb6,0xe2,0xcb,0x22,
        0x29,0xde,0x28,0xb8,0x5c,0xc3,0xd8,0x38,
        0x2d,0xbd,0xd9,0xd1,0xa1,0xcf,0xb2,0xd0,
        0xe1,0xc1,0xe0,0xce,0xc7,0xb5,0xdc,0xb7,
        0xc0,0xd5,0xbc,0xc8,0xcc,0xda,0xca,0xd4,
        0xb9,0xdd,0xba,0xae,0x7c,0x2c,0x25,0x7f};
        extern void far welon();
        extern void far weloff();
        extern void far uplmem();
        extern void far lolmem();
        extern void far wrcolor();
        extern void plchar();
/*part of display*/
        lAX = 0x0013;
        geninterrupt(0x10);
        if((fp = fopen("d:\\work\\yai\\thai14.fnt","rb")) == NULL)
        {   putch('\x007');
            exit(0);
        }
        fread(buff1,3584,1,fp);
        fclose(fp);
        prev =peekb(0xa000,offset);
        pokeb(0xa000,offset,attrib);
        while((ch = getch()) != '\n')
        {   if(ch == 0x00)
            {   ch = getch();
                switch(ch){
      case 0x4d:pokeb(0xa000,offset,prev);offset += 8;
                if((offset > 63999)||(offset < (8*320)))
                   offset = 8*320;
```

```
            prev = peekb(0xa000,offset);
            pokeb(0xa000,offset,attrib);break;
case 0x4b:pokeb(0xa000,offset,prev);offset -= 8;
            if((offset > 63999)||(offset < (8*320)))
               offset = 8*320;
            prev = peekb(0xa000,offset);
            pokeb(0xa000,offset,attrib);break;
case 0x50:pokeb(0xa000,offset,prev);offset += (28*320);
            if((offset > 63999)||(offset < (8*320)))
               offset = 8*320;
            prev = peekb(0xa000,offset);
            pokeb(0xa000,offset,attrib);break;
case 0x48:pokeb(0xa000,offset,prev);offset -= (28*320);
            if((offset > 63999)||(offset < (8*320)))
               offset = 8*320;
            prev = peekb(0xa000,offset);
            pokeb(0xa000,offset,attrib);break;
  default:break;
                  }
        }
     else  if(ch == 0x08)
        {  if(thmode == FALSE)
           {  pokeb(0xa000,offset,prev);
              offset -= 8;prev = peekb(0xa000,offset);
              for(i=0;i<14;i++)
              {  for(j=0;j<8;j++)
                 {  pokeb(0xa000,offset,0x00);
                    offset++;
                 }
                 offset = (offset-8)+320;
              }
              offset = offset-(14*320);
              pokeb(0xa000,offset,attrib);
```

```
                }
            else {
                    poke(0xa000,offset,prev);
                    offset -= ((7*320)+8);
                    for(i=0;i<28;i++)
                    {  for(j=0;j<8;j++)
                       {  pokeb(0xa000,offset,0x00);
                          offset++;
                       }
                       offset = (offset-8)+320;
                     }
                     offset = offset-(21*320);
                     prev = peekb(0xa000,offset);
                     pokeb(0xa000,offset,attrib);
                    }
            }
        else if (ch=='\r')
            {  pokeb(0xa000,offset,prev);
               enter = offset%320;
               offset += (28*320-enter);
               prev = peekb(0xa000,offset);
               pokeb(0xa000,offset,attrib);
            }
        else if(ch == ESC)
            {  thmode = (!thmode?TRUE:FALSE);
            }
        else if((ch >30)&&(ch<0x7f))
            {  if(thmode == FALSE)
               {   plchar(ch,offset);offset += 8;
                   prev = peekb(0xa000,offset);
/* english */      pokeb(0xa000,offset,attrib);
               }
/* thai   */ else { chr = thaitable[ch];
```

```
                    if((chr>0xa0)&&(chr<0xd7))
                    {  plchar(chr,offset);offset += 8;
                       prev = peekb(0xa000,offset);
                       pokeb(0xa000,offset,attrib);
/* ]]]]] */            }
                  else if((chr==0xd7)||(chr==0xd8))
                    {  pokeb(0xa000,offset,prev);
                       offset += ((14*320)-8);
                       plchar(chr,offset);offset -=((14*320)-8);
                       prev = peekb(0xa000,offset);
                       pokeb(0xa000,offset,attrib);
/* ]]]]] */            }
                  else if((chr > 0xd8)&&(chr < 0xff))
                    {  poke(0xa000,offset,prev);
                       offset -= ((14*320)+8);
                       plchar(chr,offset);offset +=((14*320)+8);
                       prev = peekb(0xa000,offset);
                       pokeb(0xa000,offset,attrib);
/* ]]]]] */            }
                }
             }
          }
          ]AX = 0x0003;
          geninterrupt(0x10);

 }


void plchar(char code,unsigned int addr)
{
   char chr[14];
   int i,j;
   char asc;
```

```
extern char prev,buff1[3584];
      i = 14*code;
      for(j=0;j<14;j++)
      chr[j] = buff1[i+j];
      for(i=0;i<14;i++)
      {   asc = chr[i];
          for(j=0;j<8;j++)
          if(asc>=0x80)
           {  pokeb(0xa000,addr,attrib);addr++;
              prev = peekb(0xa000,addr);
              asc = (asc<<1);
           }
      else {  pokeb(0xa000,addr,prev);
              addr++;
              prev =peekb(0xa000,addr);
              asc = (asc<<1);
          }
         addr = (addr-8)+320;
         prev = peekb(0xa000,addr);
      }
}
```

PROGRAM EDGE DETECTION

```
#include<stdio.h>
#include<dos.h>
#include<stdlib.h>
#include<alloc.h>

#define IE 128
#define JE 256
#define JE]AFTER]EDGE 254

char far *crt]addr;
char far *block4,*block6;
char far *block5;

edge()
{
   char indx[IE][2];
   static unsigned char ia[IE][JE];
   int i,j,val,test = 0;
   long int k = 0l;
   char name[10];
   FILE *out;
   extern void edetect();
   extern void far up]mem();
   extern void far lo]mem();

        crt]addr = 0xd0000000;
        up]mem();we]off();
        if((block4 = farmalloc(87536)) == NULL)
        {  gotoxy(23,40);
           printf("error allocate");
```

```
    exit(0);
}
block5 = block4;
for (i=0;i<IE;i++)
{  for(j=0;j<JE;j++)
   {  ia[i][j] = *crt_addr;
      crt_addr += 2; k += 2;
      if (k > 65535L)  lo_mem();
   }
   crt_addr += (1*512); k += (1*512);
}
edetect(ia);
crt_addr = 0xd0000030;
block5 = block4;
for(i=0;i<IE;i++)
{  for(j=0;j<JE_AFTER_EDGE;j++)
   {  *crt_addr = *block5;
      crt_addr++;block5++;
   }
   crt_addr += (512-JE_AFTER_EDGE);
}
for(i=0;i<IE;i++)
for(j=0;j<2;j++)
   indx[i][j] = 0x00;
for(i=0;i<IE;i++)
{  j=10;
   block5 = (block4+(i*JE_AFTER_EDGE)+j);
   while(j<JE_AFTER_EDGE)
   {  block5++;
      block6 = block5-1;
      if((val = abs((*block6)-(*block5))) < 0x30) j++;
  else {  indx[i][0] =(char)j ;k=JE-12;j = JE_AFTER_EDGE;
          while( k>(long)indx[i][0])
```

```
            {   block5 = block4+(i*JE]AFTER]EDGE)+k;
                block6 = block5-1;
               if((val = abs((*block6)-(*block5))) < 0x30) k--;
          else {indx[i][1] = (char)k;k = 0;}
            }
         }
    }
}
up]mem();
for(i=0;i<IE;i++)
{  crt]addr = 0xd0002048;
   if(indx[i][0] == 0x00) i++;
   else  crt]addr += ((i*512)+indx[i][0]);
   for(j=(int)indx[i][0];j<=(int)indx[i][1];j++)
   {  *crt]addr = ia[i][j];
      crt]addr++;
   }
}
k = 0;
block5 = block4;
for(i=0;i<IE;i++)
{  *block5 = indx[i][0];block5++;k++;
   *block5 = indx[i][1];block5++;k++;
   for(j=(int)indx[i][0];j<(int)indx[i][1];j++)
   {  if(ia[i][j] == 0x1a)  *block5 = 0x19;
     else *block5 = ia[i][j];
       block5++;k++;
   }
}
gotoxy(20,40);
printf("%c%c%c%c%c%c%c%c%c%c%cCompress  To%c%c%c%c%c%c%c%c%c%c%c",
214,196,196,196,196,196,196,196,196,196,196,196,196,
196,196,196,196,196,196,196,196,183);
```

```
        gotoxy(21,40);
        printf("%c",186);
        for(i=0;i<32;i++)
           printf("%c",32);
        printf("%c",186);
        gotoxy(22,40);
        printf("%c",211);
        for(i=0;i<32;i++)
           printf("%c",196);
        printf("%c",189);
        gotoxy(21,43);
        block5 = block4;
        gets (name) ;
        if ((out = fopen(name,"wb")) != NULL)
        {  fwrite(block5,k+1,1,out);fclose(out);farfree(block4);
           gotoxy(21,43);printf(" Save Detected Edge Complete");
        }
     else { gotoxy(21,43);printf("Error Open File");}
        getch();
        gotoxy(20,40);for(i=0;i<280;i++)printf("%c",32);

}


void edetect(char ia[IE][JE])
{
   int i,j,i1,j1,k,k1,k2,im,ip,jm,jp;
   extern char far *block4,*block6;
   extern char far *block5;

        i1 = IE-1; j1 = JE-1;
        for(i=1;i<i1;i++)
```

```
    {   im = i-1; ip = i+1;
        for(j=1;j<j1;j++)
        {  jm = j-1;jp = j+1;
    k1 = -ia[im][jm]-2*ia[im][j]-ia[im][jp]+ia[ip][jm]+2*ia[ip][j]+ia[ip][jp];
    k2 = -ia[im][jm]-2*ia[i][jm]-ia[ip][jm]+ia[im][jp]+2*ia[i][jp]+ia[ip][jp];
    k1 = k1<0?-k1:k1;
    k2 = k2<0?-k2:k2;
    k  = k1+k2;
    *block5 = k>1277127:k;
    block5++;
        }
    }
}
```

PROGRAM DISPLAY IMAGE TO VGA

```
#include<stdio.h>
#include<dos.h>


disp()
{
   int i,j,value,divide;
   char temp;
   FILE *fp;
   long k;
   char far *vga]addr,*crt]addr;
   extern void far we]on();
   extern void far we]off();
   extern void far up]mem();
   extern void far lo]mem();
   extern void far wrcolor();

        ]AX = 0x0013;
        geninterrupt(0x10);
        for(i=0;i<64;i++)
        wrcolor(i,i,i,i);
        we]on();
        lo]mem();
        crt]addr = (char far *)(0xd0000000+64*512);
        temp = *crt]addr;
        for(i=0;i<511;i++)
        {  crt]addr++;
           if(*crt]addr > temp)temp = *crt]addr;
        }
        divide = temp/64;
```

```
uplmem();
k=0;
crtladdr = (char far *)0xd0003860;
vgaladdr = (char far *)0xa0000000;
for(i=0;i<200;i++)
{  for(j=0;j<320;j++)
   {  value = (*crtladdr)/divide;
      if(value > 63) value = 63;
      *vgaladdr = value;
      crtladdr++;k++;vgaladdr++;
   }
crtladdr += 192;
if(k==32000)
{  lolmem();crtladdr=(char far *)0xd0000060;  }
}
weloff();
}
```

PROGRAM LOAD TEXT&IMAGE FILE

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
#include<alloc.h>
#include<dos.h>


#define IE 128
#define JE 256
#define attrib 0xA0
#define prev 0x05
#define TRUE  1
#define FALSE 0
#define UP    0
#define LO    1
#define ESC   0x1B


pop()
{  unsigned int i,j,dbJfp;
   int x,y,l;
   char far *crtJaddr ;
   FILE *in;
   long k = (28*512+32);
   int offset=(28*512+32);
   char source,dest;
   char *name1,*name;
   int thmode = FALSE;
   int part   = UP;
   char ch,chr;
```

```
char array[3000];
char far *array1,*array2;
extern char buff1[3584];
extern void far welon();
extern void far weloff();
extern void far uplmem();
extern void far lolmem();
extern void far cursor();
extern void plch();

    cursor(20,40);
    printf("%c%c%c%c%c%c%c%c%c%cFileName To Pop Up%c%c%c%c%c%c%c%c%c%c",
    214,196,196,196,196,196,196,196,196,196,196,196,196,196,196,196,
    196,196,196,196,183);
    cursor(21,40);
    printf("%c",186);
    for(i=0;i<36;i++)
         printf("%c",32);
    printf("%c",186);
    cursor(22,40);
    printf("%c",211);
    for(i=0;i<36;i++)
         printf("%c",196);
    printf("%c",189);
    if((array1 = farmalloc(32000)) == NULL)
    {  cursor(23,40);
       printf("error allocate");
       exit(0);
    }
    array2 = array1;
    cursor(21,43);
    gets(name1);
    if((in = fopen(name1,"rb")) == NULL)
```

```
    {  cursor(20,45);
       for(i=0;i<26;i++)
          printf("%c",196);
       cursor(21,43);
       printf("     Error  Open  File    ");
       exit(0);
    }
 else {  fread(array1,32000,1,in);
         fclose(in);
      }
    uplmem();
    weloff();
    for(i=0;i<IE;i++)
    {  crtladdr = 0xd0000000+256+(i*512);
       for(j=0;j<JE;j++)
       {  *crtladdr = 0x66;crtladdr++;}
    }
    for(i=0;i<IE;i++)
    {  source = *array1;array1++;
       dest   = *array1;array1++;
       crtladdr = (0xd0000000+256)+(i*512)+source;
       for(j=source;j<dest;j++)
       {   *crtladdr = *array1;
           crtladdr++;array1++;
       }
    }
    welon();
    farfree(array2);
    cursor(20,45);
    for(i=0;i<26;i++)
        printf("%c",196);
    cursor(21,43);printf("Load Compressed Image Complete");
    if((in = fopen("d:\\work\\yai\\thai14.fnt","rb")) == NULL)
```

```
{  putch('\x007'); exit(0);}
fread(buff1,3584,1,in);fclose(in);
if((in = fopen("d:\\work\\boon\\dbase\\image.dbf","rb")) != NULL)
{  fread(array,3000,1,in);
   fclose(in);
}
db]fp = 194;y=0;
l = strlen(name1);
name = &array[db]fp];
while(strncmp(name1,name,l))
{  db]fp += 130;y++;
   name = &array[db]fp];
   if(y==50){ cursor(20,45);
              for(i=0;i<26;i++)
              printf("%c",196);
              cursor(21,43);
              printf("Can't Find This Filename\n");
              exit(0);
            }
}
we]off();
offset = (40*512+40);
db]fp = 204+(y*130);
for(x=0;x<5;x++)
{  p]ch(array[63+x],offset);offset+=8; }
for(x=0;x<22;x++)
{   p]ch(array[db]fp],offset);
    db]fp++; offset+=8;
}
offset = (75*512)+40;
for(x=0;x<10;x++)
{  p]ch(array[95+x],offset);offset+=8; }
offset +=24;
```

```
    for(x=0;x<6;x++)
    {  plch(array[dbfp],offset);dbfp++;
       offset+=8;
    }
    offset = (110*512)+40;
    for(x=0;x<8;x++)
    {  plch(array[127+x],offset);
       offset+=8;
    }
    lolmem();
    offset = (15*512)+128;
    for(x=0;x<40;x++)
    {  plch(array[dbfp],offset);
       dbfp++;offset +=8;
    }
    offset = (52*512)+128;
    for(x=0;x<40;x++)
    {  plch(array[dbfp],offset);
       dbfp++;offset+=8;
    }
    offset = (90*512)+40;
    for(x=0;x<10;x++)
    {  plch(array[159+x],offset);
       offset+=8;
    }
    offset +=24;
    for(x=0;x<11;x++)
    {  plch(array[dbfp],offset);dbfp++;
       offset+=8;
    }
    welon();
    getch();
}
```

```
void plch(char code,unsigned int addr)
{  char chr[14];
   int i,j;
   char asc;
   extern char buff1[3584];
        i = 14*code;
        for(j=0;j<14;j++)
              chr[j] = buff1[i+j];
        for(i=0;i<14;i++)
        {   asc = chr[i];
            for(j=0;j<8;j++)
                if(asc>=0x80)
                {  pokeb(0xd000,addr,attrib);addr++;
                   asc = (asc<<1);}
               else { pokeb(0xd000,addr,prev);
                      addr++;
                      asc = (asc<<1);
                    }
         addr = (addr-8)+512;
       }
}
```

PROGRAM CLEAR MONITOR

```
PROG    SEGMENT
        PUBLIC  ]clr]crt
        ASSUME  CS:PROG

]clr]crt  PROC  far
          push bp
          mov  bp,sp
          push es
          push di
          pushf
          mov  al,00h          ;up]mem()
          mov  dx,2f0h
          out  dx,al
          mov  al,00h          ;we]off()
          mov  dx,2f1h
          out  dx,al

          cld
          mov ax,0D000h
          mov es,ax
          mov al,05h
          mov di,0000h
          mov cx,0FFFFh
          rep stosb
          mov  al,01h          ;lo]mem()
          mov  dx,2f0h
          out  dx,al

          mov ax,0D000h
          mov es,ax
```

```
                mov al,05h
                mov di,0000h
                mov cx,0FFFFh
                rep stosb

                mov  al,02h             ;welon()
                mov  dx,2f1h
                out  dx,al
                popf
                pop di
                pop es
                pop bp
                ret
JclrJcrt        ENDP
PROG            ENDS
                END
```

PROGRAM TOOLXT.ASM

```
PGROUP   GROUP    PROG
PROG     SEGMENT  BYTE PUBLIC 'CODE'
         PUBLIC   ]up]mem,]lo]mem,]we]on,]we]off,]freeze,]no]frz,]re]on,]re]o
         ASSUME   CS:PGROUP

]up]mem proc  far
        mov   al,00h
        mov   dx,2f0h
        out   dx,al
        ret
]up]mem endp

;-----------------------------------------
]lo]mem proc  far
        mov   al,01h
        mov   dx,2f0h
        out   dx,al
        ret
]lo]mem endp
;-----------------------------------------
]we]off proc  far
        mov   al,00h
        mov   dx,2f1h
        out   dx,al
        ret
]we]off endp
;-------------------------------------
]we]on  proc  far
        mov   al,02h
        mov   dx,2f1h
        out   dx,al
```

```
        ret
]weJon  endp
;---------------------------------------
]freeze proc  far
        mov   al,00
        mov   dx,2f1h
        out   dx,al
        ret
]freeze endp
;---------------------------------------
]noJfrz proc  far
        mov   al,01
        mov   dx,2f1h
        out   dx,al
        ret
]noJfrz endp
;---------------------------------------
]reJon  proc  far
        mov   al,02
        mov   dx,2f0h
        out   dx,al
        ret
]reJon  endp
;---------------------------------------
]reJof  proc  far
        mov   al,00
        mov   dx,2f0h
        out   dx,al
        ret
]reJof  endp
;---------------------------------------
        ;pokebV (segment,offset,value)
```

```
]pokebV proc  far
        push  bp
        mov   bp,sp
        mov   ax,[bp+6]
        mov   es,ax
        mov   bx,[bp+8]
        mov   al,[bp+12]
        mov   es:[bx],al
        mov   sp,bp
        pop   bp
        ret
]pokebV endp
;---------------------------------------
        ;peekbV (segment,offset)
]peekbV proc  far
        push  bp
        mov   bp,sp
        mov   ax,[bp+6]
        mov   es,ax
        mov   bx,[bp+8]
        mov   al,es:[bx]
        mov   sp,bp
        pop   bp
        ret
]peekbV endp
;---------------------------------------
]cr]mon proc  far
        push  bp
        mov   bp,sp
        push  es
        push  di
        pushf
        mov   al,00h            ;up]mem()
```

```
        mov   dx,2f0h
        out   dx,al
        mov   al,00h            ;weloff()
        mov   dx,2f1h
        out   dx,al
        cld
        mov   ax,0D000h
        mov   es,ax
        mov   al,05h
        mov   di,0000h
        mov   cx,0FFFFh
        rep   stosb
        mov   al,01h            ;lolmem()
        mov   dx,2f0h
        out   dx,al
        mov   ax,0D000h
        mov   es,ax
        mov   al,05h
        mov   di,0000h
        mov   cx,0FFFFh
        rep   stosb
        mov   al,02h            ;welon()
        mov   dx,2f1h
        out   dx,al
        popf
        pop   di
        pop   es
        pop   bp
        ret
lcrlmon endp
;---------------------------------------
PROG    ENDS
        END
```

PROGRAM VGATOOL.ASM

```
PGROUP    GROUP     PROG
PROG      SEGMENT   BYTE PUBLIC 'CODE'
          PUBLIC    JsetVGA,Jgray,Jwrcolor,Jsetcur,Jtran,Jsetpal
          ASSUME    CS:PGROUP
;-----------------------------------------
        ;setVGA (mode)
JsetVGA   proc      far
          push      bp
          mov       bp,sp
          mov       ah,00H
          mov       al,[bp+6]
          int       10H
          mov       sp,bp
          pop       bp
          ret
JsetVGA   endp
;-----------------------------------------
   ;setpal (regnum,value)
Jsetpal   proc      far
          push      bp
          mov       bp,sp
          mov       bl,[bp+6]
          mov       bh,[bp+8]
          mov       ax,1000H
          int       10H
          mov       sp,bp
          pop       bp
          ret
Jsetpal   endp
;-----------------------------------------
```

```
 ;gray (buffer,firstreg,numreg)
]gray    proc      far
         push      bp
         mov       bp,sp
         les       dx,[bp+6]
         mov       bx,[bp+10]
         mov       cx,[bp+12]
         mov       ax,1012H
         int       10H
         mov       sp,bp
         pop       bp
         ret
]gray    endp
;-----------------------------------------
  ;wrcolor (regnum,red,green,blue)
]wrcolor proc      far
         push      bp
         mov       bp,sp
         mov       bx,[bp+6]       ;REGNUM
         mov       dh,[bp+8]       ;RED
         mov       ch,[bp+10]      ;GREEN
         mov       cl,[bp+12]      ;BLUE
         mov       ax,1010H
         int       10H
         mov       sp,bp
         pop       bp
         ret
]wrcolor endp
;-----------------------------------------
         ;setcur (row,col)
]setcur  proc      far
         push      bp
         mov       bp,sp
```

```
        mov     ah,02H
        mov     ch,[bp+6]  ;ROW
        mov     dl,[bp+8]  ;COLOUMN
        mov     bh,[bp+10] ;page
        int     10H
        mov     sp,bp
        pop     bp
        ret
lsetcur endp
;-----------------------------------------
ltran   proc    far
        mov     al,00h     ;uplmem
        mov     dx,2f0h
        out     dx,al

        push    ds
        mov     ax,0d000H
        mov     ds,ax
        mov     si,0000H
        mov     di,si
        mov     ax,0a000H
        mov     es,ax
        mov     cx,0FFFFH
        rep     movsb
        pop     ds

        mov     al,01h     ;lolmem
        mov     dx,2f0h
        out     dx,al

        push    ds
        mov     ax,0d000H
        mov     ds,ax
```

```
        mov     si,0000H
        mov     di,si
        mov     ax,0a000H
        mov     es,ax
        mov     cx,0FFFFH
        rep     movsb
        pop     ds

        ret
Jtran   endp

PROG    ENDS
        END
```

กิตติกรรมประกาศ

วิทยานิพนธ์เรื่อง คอมพิวเตอร์กราฟฟิก นี้ สำเร็จลุล่วงไปได้ด้วยดี เพราะได้รับคำปรึกษาแนะนำ และช่วยเหลือจาก รศ. มนัส สังวรศิลป์ , อาจารย์สุรพันธุ์ เอื้อไพบูลย์ , พี่อ้อย, พี่เปี๊ยก ตลอดจนพี่ๆ เพื่อนๆทุกคนที่ให้กำลังใจมาโดยตลอด จึงขอแสดงความขอบคุณมา ณ. ที่นี้ด้วย

หนังสืออ้างอิง

1. Richard F.Ferraro,"Programmer's Guide to the EGA and VGA Card",Addison-wesley Publishing Company,Inc.,606 p.,1988.

2. Bradley Dyck Kliewer,"EGA/VGA A Programmer's Reference Guide", McGraw-Hill Book Company,269 p.,1988.

3. Anil K. Jain,"Fundamentals of Digital Image Processing" ,Prentice-Hall International,Inc.,569 p.,1989.

4. Yoram Koren,"Robotics for Engineers",McGraw-Hill Book Company,1987.