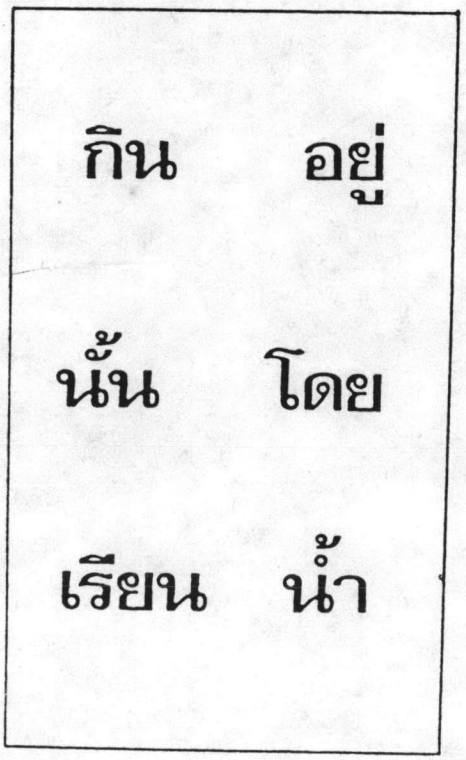
บทที่ 3

การตรวจรู้อักขระภาษาไทย

3.1 โครงสร้างของอักขระและคำในภาษาไทย

อักขระภาษาไทยนั้นประกอบด้วย พยัญชนะ 44 รูป (ปัจจุบันเลิกใช้ไป 2 รูป คือ ฃ ฅ) สระ 21 รูป และวรรณยุกต์อีก 4 รูป ถ้ามองอักขระภาษาไทยในลักษณะของรูปทรงทางเรขาคณิตจะประกอบด้วยรูป วงรี วงกลม สามเหลี่ยม สี่เหลี่ยมและกากบาท(ตัวอย่างของอักขระภาษาไทยได้แสดงเอาไว้ในรูปที่ 1.2) ในภาษาไทยนั้นจะแบ่งตำแหน่งการจัดเรียงพยัญชนะ สระ และวรรณยุกต์ออกเป็น 4 ระดับ โดยที่ระดับที่ 1 และ 2 จะเป็นตำแหน่งของสระ และวรรณยุกต์ ระดับที่ 3 จะเป็นตำแหน่งของสระ และพยัญชนะ ส่วนในระดับที่ 4 จะเป็นตำแหน่งของสระเพียง 2 รูปคือ ุ และ ู ตัวอย่างของการจัดเรียง พยัญชนะ สระและวรรณยุกต์ให้เป็นคำในภาษาไทยได้แสดงไว้ในรูปที่ 3.1



รูปที่ 3.1 แสดงคำในภาษาไทย

3.2 สมาชิกของภาพบิทเมทริกซ์ตัวอักษร

อักขระภาษาไทยที่ใช้ในการตรวจรู้นี้จะมีลักษณะเป็นภาพบิทเมทริกซ์ตัวอักษรขนาด 20 X 20 หน่วย โดยที่สมาชิกของภาพบิทเมทริกซ์ตัวอักษรนี้จะประกอบด้วยค่า 0 และค่า 1 เท่านั้น ค่าทั้งสองนี้จะขึ้นอยู่กับการแปลงสีหรือความเข้มของตัวอักขระด้วย A/D Converter กำหนดให้ค่า 0 แทนบริเวณที่เป็นพื้นที่สีขาวของเนื้อกระดาษ และค่า 1 แทนบริเวณที่เป็นหมึกพิมพ์ของอักขระแต่ละรูปในเอกสาร

3.3 การแปลงภาพบิทเมทริกซ์ตัวอักษรให้เป็นโครงร่าง (Skeleton)[4]

ในการตรวจรู้ตัวอักขระนั้นปัญหาหนึ่งที่ต้องประสบอยู่เสมอ คือ ขนาดความหนาบางของตัวอักขระที่จะตรวจรู้นั้นมีขนาดไม่เท่ากัน ซึ่งจะทำให้รหัสที่ใช้แทนอักขระรูปเดียวกันแต่มีความหนาบางต่างกันเป็นรหัสที่แตกต่างกันออกไปด้วย เพื่อตัดปัญหานี้เราจะใช้วิธีการแปลงภาพบิทเมทริกซ์ตัวอักษรที่จะตรวจรู้นั้นให้เหลือเพียงโครงร่างที่มีความหนาบางเท่ากันตลอดไม่ว่าตัวอักขระเหล่านั้นจะมีความหนาบางเริ่มแรกเท่าใดก็ตาม

การแปลงอักขระให้เหลือเพียงโครงร่างนั้นทำได้โดยการแปลงค่า 1 ทุกค่าในภาพบิทเมทริกซ์ตัวอักษรที่ทำให้อักขระนั้นมีความหนากว่าอักขระปกติให้มีค่าเป็น 0 โดยไม่ทำให้โครงร่างอักขระนั้นเปลี่ยนไป การพิจารณาว่าค่า 1 ในภาพบิทเมทริกซ์ตัวอักษรค่าใดสามารถจะเปลี่ยนให้มีค่าเป็น 0 ได้หรือไม่นั้นจะขึ้นอยู่กับค่า 0 และ 1 บริเวณรอบๆ ค่า 1 ที่กำลังพิจารณาอยู่

การแปลงภาพบิทเมทริกซ์ตัวอักษรให้เป็นโครงร่างนั้นหากเปลี่ยนค่า 1 ให้มีค่าเป็น 0 ทันทีจะทำให้การพิจารณาค่า 1 ในบริเวณใกล้เคียงกับค่า 1 ที่เปลี่ยนเป็นค่า 0 นั้นเกิดความผิดพลาดได้ เพื่อหลีกเลี่ยงความผิดพลาดดังกล่าวจึงกำหนดให้มีการเปลี่ยนค่า 1 ให้เป็นค่า 0 แบบชั่วคราวและแบบถาวรโดยใช้ค่า 7 แทนค่า 1 ที่เปลี่ยนเป็นค่า 0 แบบชั่วคราว และใช้ค่า 8 หรือค่า 9 แทน

ค่า 1 ที่เปลี่ยนเป็นค่า 0 แบบถาวร การที่จะใช้ค่า 8 หรือค่า 9 แทนการเปลี่ยนค่าแบบถาวรนี้ขึ้นอยู่กับทิศทางของการพิจารณาเปลี่ยนค่า 1 ให้เป็นค่า 0 นั้น

ทิศทางการพิจารณาเปลี่ยนค่า 1 ให้เป็นค่า 0 มีอยู่สองทิศทางคือ การพิจารณาค่า 1 จากด้านขวาไปด้านซ้าย และการพิจารณาค่า 1 จากด้านซ้ายไปด้านขวาของภาพบิทเมทริกซ์ตัวอักษร การพิจารณาค่า 1 ทั้งสองทิศทางนี้จะพิจารณา จากแถวบนสุดลงสู่แถวล่างสุดภายในภาพบิทเมทริกซ์ตัวอักษร สาเหตุที่ต้องแบ่งการพิจารณาเปลี่ยนค่า 1 ให้เป็น 0 ออกเป็นสองทิศทางก็เพื่อความสมดุลย์ของโครงร่างของอักขระที่ได้ (5)

หลักการแปลงภาพบิทเมทริกซ์ตัวอักษรให้เหลือเพียงโครงร่างมีดังนี้

3.3.1 กำหนดให้สมาชิกของภาพบิทเมทริกซ์ตัวอักษรที่กำลังพิจารณาอยู่มีค่าเป็น N_0 และให้ค่าของสมาชิกอื่นๆ ที่อยู่รอบค่า N_0 นั้นมีค่าเป็น N_1 ถึง N_8 ตามลำดับ (รูป 3.2)

3.3.2 จะพิเคราะห์ (Scan) ค่า N_0 ที่มีค่าเป็น 1 เท่านั้น

3.3.3 ให้ N_0 ทุกค่าที่มีตำแหน่งอยู่ที่ขอบทั้งสี่ด้านของภาพบิทเมทริกซ์ตัวอักษรมีค่าเป็น 0 ทั้งหมด

N_4	N_3	N_2
N_5	N_0	N_1
N_6	N_7	N_8

รูปที่ 3.2 แสดงตำแหน่งของ N_0 ถึง N_8

3.3.4 การพิเคราะห์ค่า N_0 จะพิเคราะห์จากด้านขวาไปด้านซ้ายมีหลักในการพิเคราะห์ตามแบบการคำนวณในรูปที่ 3.3 ดังนี้

```
BEGIN
  FOR  row := 2 TO 19 DO
  BEGIN
    FOR col := 19 TO 2 DO
    BEGIN
    (*  set No - N8 value  *)
    WHILE (No=1) DO BEGIN
      if (N1<>0) or (N5<>0) then
      if (N3<>0) or (N7<>0) then
      if (N6=0) and (N8=1)  then No := 7;
                            else No := 9
                    END;
    END;
  END
END.
```

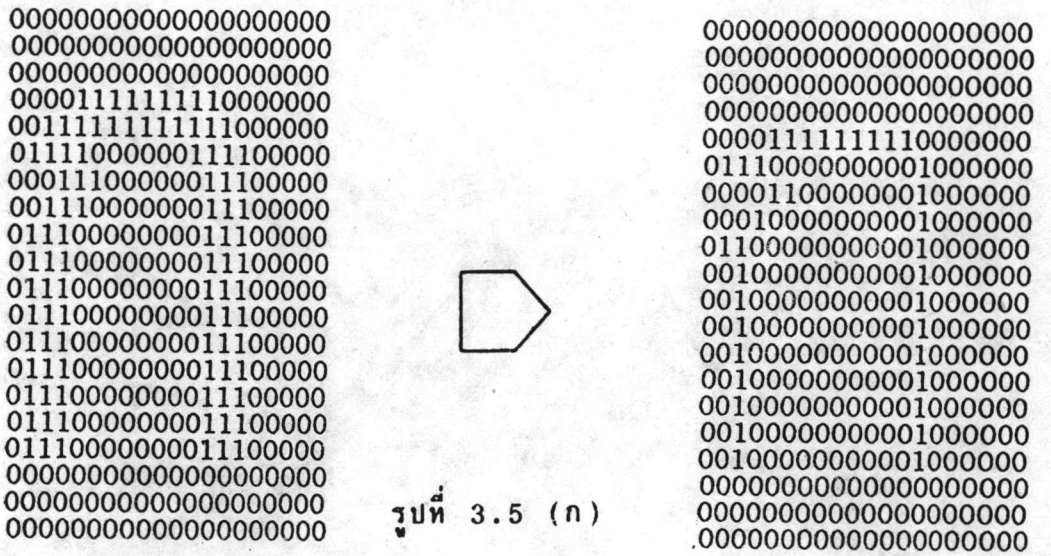
รูปที่ 3.3 แสดงแบบการคำนวณที่ใช้พิเคราะห์ค่า N_0 จากด้านขวาไปด้านซ้าย

3.3.5 การพิเคราะห์ค่า N_0 จะพิเคราะห์จากด้านซ้ายไปด้านขวา มีหลักในการพิเคราะห์ตามแบบการคำนวณในรูปที่ 3.4 ดังนี้

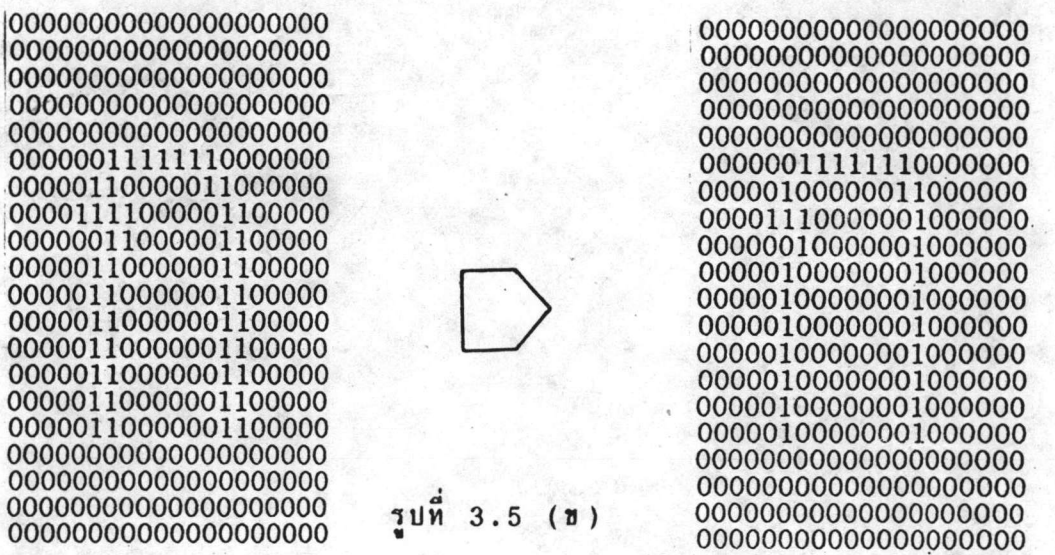
```
BEGIN
  FOR  row := 2 TO 19 DO
  BEGIN
    FOR col := 2 TO 19 DO
    BEGIN
    (*  set No - N8 value  *)
    WHILE (No=1) DO BEGIN
      if (N1<>9) then No := 8;
      (* change all No:= 8 that not
         damage character skeleton *)
                    END;
    END;
  END
END.
```

รูปที่ 3.4 แสดงแบบการคำนวณที่ใช้พิเคราะห์ค่า N_0 จากด้านซ้ายไปด้านขวา

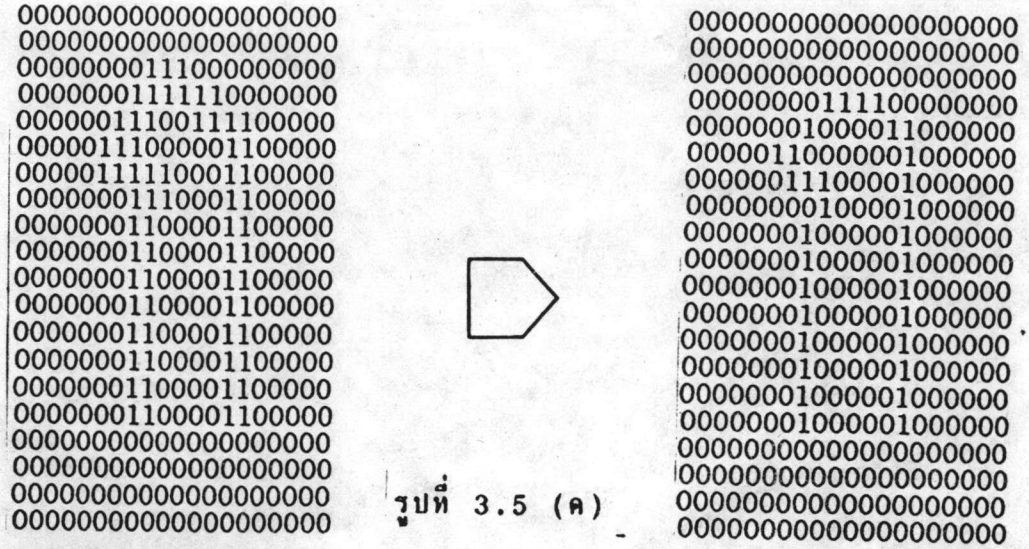
3.3.6 หลังจากพิเคราะห์ค่า N_0 ทุกค่าภายในภาพบิทเมทริกซ์ตัวอักษรแล้วสมาชิกของภาพบิทเมทริกซ์ตัวอักษรที่มีค่าเป็น 7, 8 และ 9 จะถูกเปลี่ยนให้มีค่าเป็น 1, 0 และ 0 ตามลำดับ (ตัวอย่างของภาพบิทเมทริกซ์ตัวอักษรก่อนและหลังแปลงให้เป็นโครงร่างตัวอักษรแสดงในรูป 3.5(ก)-3.5(จ))



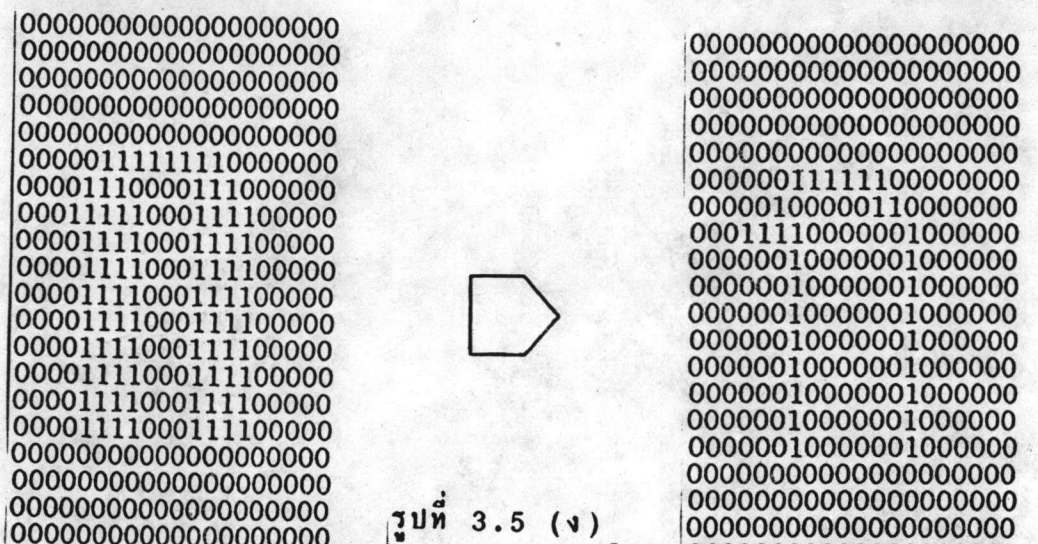
รูปที่ 3.5 (ก)



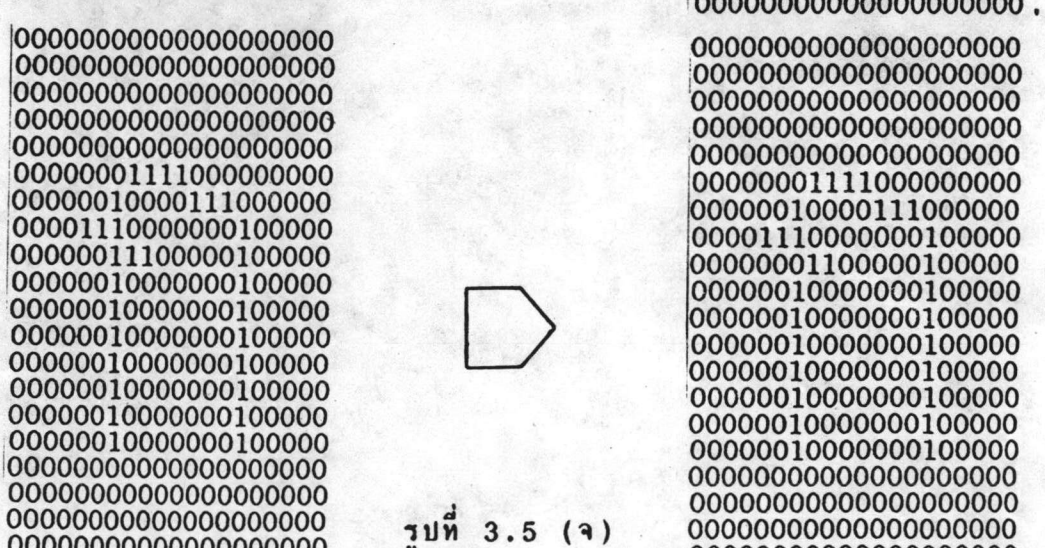
รูปที่ 3.5 (ข)



รูปที่ 3.5 (ค)



รูปที่ 3.5 (ง)



รูปที่ 3.5 (จ)

รูปที่ 3.5

แสดงภาพบิทเมทริกซ์ตัวอักษร "ก" ก่อนและหลังแปลงให้เหลือเพียงโครงร่าง

3.4 การกำหนดรหัสให้อักขระแต่ละรูป (6)

หลังจากที่ได้โครงร่างของอักขระแต่ละรูปแล้ว จะนำโครงร่างของอักขระเหล่านั้นมาตรวจรู้ด้วยการกำหนดรหัสตามแนวแถว และแนวสดมภ์ เนื่องจากอักขระแต่ละรูปจะประกอบด้วยรูปทรงทางเรขาคณิต และจำนวนรูปทรงที่ต่างกัน เมื่อแปลงอักขระเหล่านี้มาเป็นภาพบิทเมทริกซ์ตัวอักษรการกระจายของค่า 1 ภายในภาพบิทเมทริกซ์ตัวอักษรของอักขระแต่ละรูปก็จะแตกต่างกันออกไป เพราะฉะนั้นเมื่อพิเคราะห์หาจำนวนค่า 1 ตามแนวแถวและตามแนวสดมภ์ก็สามารถจะกำหนดรหัสที่ไม่ซ้ำกันของอักขระแต่ละรูปได้ ดังนี้

กำหนดให้

H = จำนวนแถวหรือสดมภ์ของภาพบิทเมทริกซ์ตัวอักษร (ในที่นี้มีค่าเท่ากับ20)

$t_1 = H/5$

$t_2 = (H/2)-1$

d = จำนวนค่า 1 ที่อยู่ในตำแหน่งที่ติดกันภายในภาพบิทเมทริกซ์ตัวอักษร

ให้ $$COD = \begin{cases} \text{small} & \text{เมื่อ} \quad d =< t_1 \\ \text{medium} & \text{เมื่อ} \quad t_1 < d =< t_2 \\ \text{long} & \text{เมื่อ} \quad d > t_2 \end{cases}$$

เมื่อหาค่า COD ของแต่ละแถวและ สดมภ์ของภาพบิทเมทริกซ์ตัวอักษรได้แล้วจะใช้ตารางที่ 3.1 ช่วยในการกำหนดรหัสตามแนวแถว (RCOD) และรหัสตามแนวสดมภ์ (CCOD) รหัสในตารางที่ 3.1 นี้ได้มาจากการพิจารณาค่าความหนาแน่นและการกระจายของค่า 1 ในภาพบิทเมทริกซ์ตัวอักษรตามแนวแถวและแนวสดมภ์โดยที่สัญญลักษณ์ที่ใช้แทนรหัสเหล่านี้จะใช้สัญญลักษณ์ใดก็ได้ แต่เพื่อความสะดวกในการประมวลผลในที่นี้จึงใช้อักษร A - P และเครื่องหมาย * แทนสัญญลักษณ์ของรหัสดังกล่าวข้างต้น

จากค่า RCOD และ CCOD ที่ได้จะนำมาย่อให้เป็นรหัสย่อตามแนวแถว (RCODC) และรหัสย่อตามแนวสดมภ์ (CCODC) โดยการดึงเอารหัสที่เหมือนกันและอยู่ในตำแหน่งที่ติดกันของรหัส RCOD และรหัส CCOD ออกมาเพียงค่าเดียว ดังตัวอย่าง

สมมุติให้ RCOD = "AAACBBDDDCC"

จะได้ RCODC = "ACBDC"

ดังนี้เป็นต้น

Combination of COD			Code
Small	-	-	A
Medium	-	-	B
Long	-	-	C
Small	Small	-	D
Medium	Small	-	E
Long	Small	-	F
Small	Medium	-	G
Medium	Medium	-	H
Long	Medium	-	I
Small	Long	-	J
Medium	Long	-	K
Small	Small	Small	M
Medium	Small	Small	N
Small	Medium	Small	N
Small	Small	Medium	N
Long	Small	Small	O
Small	Long	Small	O
Small	Small	Long	O
Medium	Medium	Small	P
Medium	Small	Medium	P
Small	Medium	Medium	P
Four CODs and above			*

ตารางที่ 3.1 ตารางกำหนดรหัสสำหรับการตรวจรู้ตัวอักขระภาษาไทย

เมื่อนำรหัส RCODC และ รหัส CCODC มาต่อกันจะได้รหัสรวม (COMCOD) ซึ่งเป็นรหัสที่ใช้ในการตรวจรู้อักขระแต่ละรูปได้ เพราะรหัสรวมของอักขระแต่ละรูปจะต่างกัน ตัวอย่างการหาค่ารหัสรวมจากโครงร่างของอักขระ "ก" (รูปที่ 3.6) มีดังนี้

```
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00001111111110000000
01110000000001000000
00001100000001000000
00010000000001000000
01100000000001000000
00100000000001000000
00100000000001000000
00100000000001000000
00100000000001000000
00100000000001000000
00100000000001000000
00100000000001000000
00100000000001000000
00000000000000000000
00000000000000000000
00000000000000000000
```


รูปที่ 3.6 แสดงโครงร่างของอักขระ "ก"

จากโครงร่างของอักขระ "ก" สามารถหาค่า COD ตามแนวแถวและแนวสดมภ์ได้ดังนี้

แถวที่ 5	COD = medium
แถวที่ 6-17	COD = small, small
สดมภ์ที่ 2	COD = small, small
สดมภ์ที่ 3	COD = small, medium
สดมภ์ที่ 4-6	COD = small, small
สดมภ์ที่ 7-13	COD = small
สดมภ์ที่ 14	COD = long

จากค่า COD ตามแนวแถว และแนวสดมภ์สามารถแปลงให้เป็นรหัส RCOD และCCOD โดยอาศัยตารางที่ 3.1 ช่วยได้ดังนี้

RCOD = "BDDDDDDDDDDDD"

CCOD = "DGDDDAAAAAAAC"

ให้ค่า RCODC และ CCODC เป็นรหัสย่อของรหัส RCOD และ CCOD ตามลำดับมีค่าดังนี้

RCODC = "BD"

CCODC = "DGDAC"

เมื่อนำรหัส RCODC และ CCODC มาต่อกันจะได้เป็นรหัสรวม (COMCOD) ดังนี้

COMCOD = "BDDGDAC"

ในทำนองเดียวกันเราสามารถจะหาค่า COMCOD ของโครงร่างอักขระ "ก" ในรูป 3.5(ก) - 3.5(จ) ได้ดังนี้

อักขระ "ก" ในรูป	COMCOD
3.5 (ก)	"BDDGDAC"
3.5 (ข)	"BDAGDAC"
3.5 (ค)	"ADANDAC"
3.5 (ง)	"BDAGAB"
3.5 (จ)	"ADAGDAB"

จากค่ารหัสรวมของอักขระ "ก" ทั้งห้าแบบพิมพ์จะสังเกตได้ว่าไม่มีค่ารหัสรวมของอักขระใดที่เท่ากัน เนื่องจากอักขระทั้งห้าแบบพิมพ์มีลักษณะของโครงร่างที่แตกต่างกัน เพราะฉะนั้นจึงต้องเก็บรหัสรวมของอักขระ "ก" ทั้งห้าแบบพิมพ์นี้ไว้ในสื่อข้อมูล เพื่อใช้เป็นรหัสต้นแบบสำหรับเปรียบเทียบกับรหัสรวมของอักขระที่ต้องการจะตรวจรู้ต่อไป

3.5 การค้นหา (Search) รหัสต้นแบบเพื่อการตรวจรู้

การค้นหารหัสต้นแบบให้มีประสิทธิภาพจะต้องคำนึงถึงองค์ประกอบที่เป็นทั้ง Hardware และ Software ดังต่อไปนี้

3.5.1 สื่อข้อมูลที่ใช้ในการประมวลผล

การวิจัยนี้ใช้เครื่องไมโครคอมพิวเตอร์ 16 บิทที่มีหน่วยความจำขนาด 512 KB (524,288 ตัวอักษร) ในการประมวลผล โดยจะใช้หน่วยความจำหลักของเครื่องคอมพิวเตอร์เป็นสื่อเก็บข้อมูลต่างๆ ขณะประมวลผล ซึ่งรายละเอียดของข้อมูลที่ต้องเก็บไว้ประมวลผลในหน่วยความจำหลักมีดังนี้

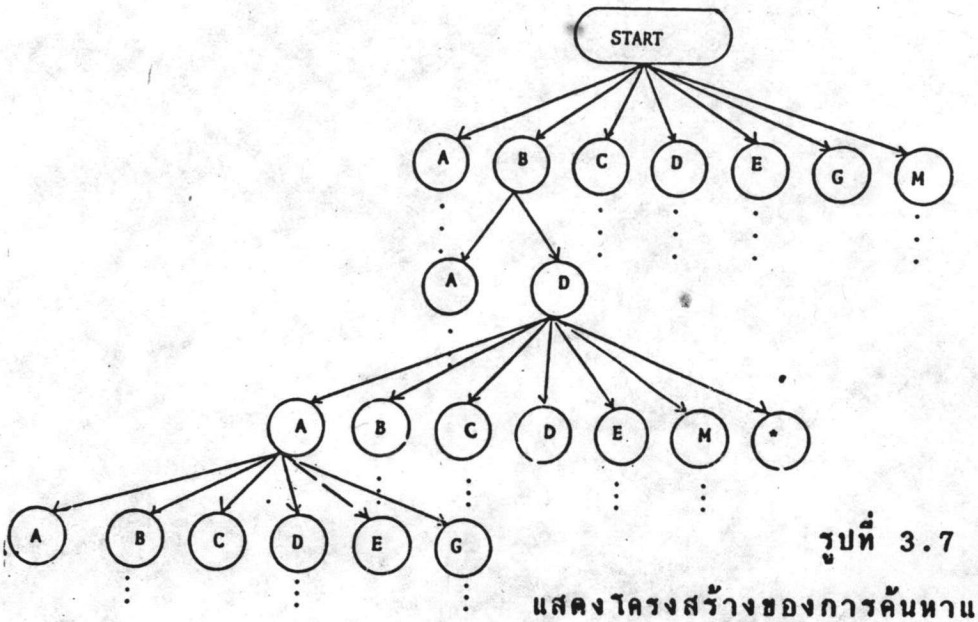
No.	Contents	Spaces (bytes)
1	Operating System (MS-DOS)	40,192
2	BASIC Runtime Module (BASRUN.EXE)	31,744
3	Recognition Program	12,288
4	Recognition Data	1,322
5	Character Fonts	16,512
6	Index File	24,320
7	Character File	12,544
	*** Total	138,922 ***

ตารางที่ 3.2 ตารางแสดงขนาดของข้อมูลที่บรรจุในหน่วยความจำหลัก

3.5.2 เทคนิคการค้นหาข้อมูล (Searching Method)

เทคนิคที่ใช้ในการค้นหาข้อมูลด้วยเครื่องคอมพิวเตอร์มีอยู่หลายวิธีเช่น การค้นหาข้อมูลแบบเรียงลำดับ (Sequential Search) การค้นหาข้อมูลแบบแบ่งครึ่งข้อมูล (Binary Search) หรือการค้นหาข้อมูลแบบกิ่งต้นไม้ (Tree Search) เป็นต้น

การวิจัยนี้จะใช้การค้นหาข้อมูลแบบกิ่งต้นไม้ เพราะเป็นการค้นหาข้อมูลที่ใช้เนื้อที่เก็บข้อมูลไม่มากนักและ ความเร็วของการค้นหาก็เร็วพอสมควรโครงสร้างกิ่งต้นไม้ที่ใช้ในการค้นหาข้อมูลมีดังนี้



รูปที่ 3.7
แสดงโครงสร้างของการค้นหาแบบกิ่งต้นไม้

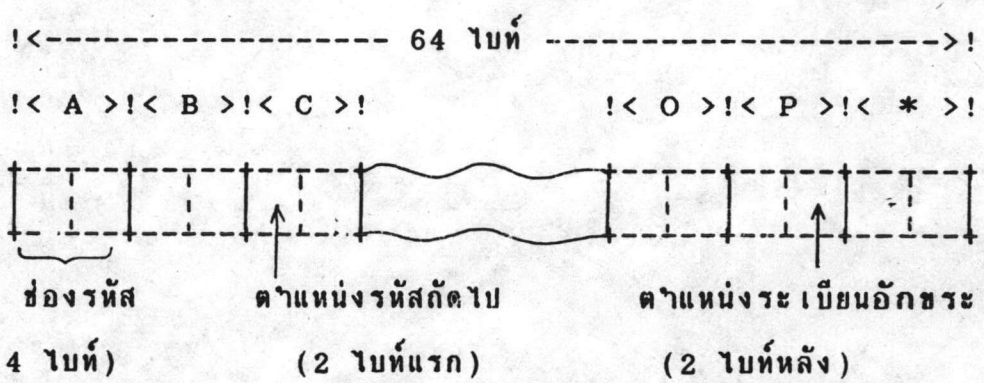
จะสังเกตได้ว่ารหัสผลลัพธ์ที่ได้จากตารางที่ 3.1 จะมีค่าทั้งหมด 16 ค่าคือ "A","B","C","D","E","F","G","H","I","J","K","M","N","O","P" และ "*" เพราะฉะนั้นกิ่งของต้นไม้ที่สร้างแต่ละกิ่งจะแตกกิ่งก้านออกไปได้สูงสุดเพียง 16 กิ่งเท่านั้น (รูปที่ 3.7) เมื่อนำรหัสรวมของอักขระแต่ละรูปมาสร้างเป็นกิ่งต้นไม้ ลำดับชั้น (Level) ของกิ่งต้นไม้จะเท่ากับจำนวนรหัสในรหัสรวมนั้นๆ แต่เพื่อความประหยัดเนื้อที่ที่ใช้เก็บข้อมูลของกิ่งต้นไม้จึงเก็บข้อมูลเฉพาะกิ่งที่มีกิ่งย่อยแตกออกไปมากกว่า 2 กิ่ง หากกิ่งใดมีกิ่งย่อยเพียงกิ่งเดียวในทุกๆ ลำดับชั้น แสดงว่ารหัสหลังจากกิ่งนั้นจะไม่มีรหัสอื่นซ้ำแล้ว จึงไม่จำเป็นต้องเสียเนื้อที่สำหรับเก็บข้อมูลเหล่านั้น

แฟ้มข้อมูล (File) ต่างๆ ที่ใช้เก็บข้อมูลของกิ่งต้นไม้มีดังนี้

3.5.2.1 แฟ้มข้อมูลดัชนี (Index File)

เป็นแฟ้มข้อมูลที่ใช้เก็บตำแหน่งระเบียน (Record) ของรหัสตัวถัดไป และ ตำแหน่งที่เก็บรายละเอียดของรหัสที่ค้นหา ระเบียนในแฟ้มข้อมูลดัชนีทุกระเบียนยาว 64 ไบท์ (Bytes) ใน 64 ไบท์นี้จะแบ่งเป็น 16 ช่องรหัส ตำแหน่งของแต่ละช่องรหัสจะแทนด้วยรหัสผลลัพธ์ที่ได้จาก

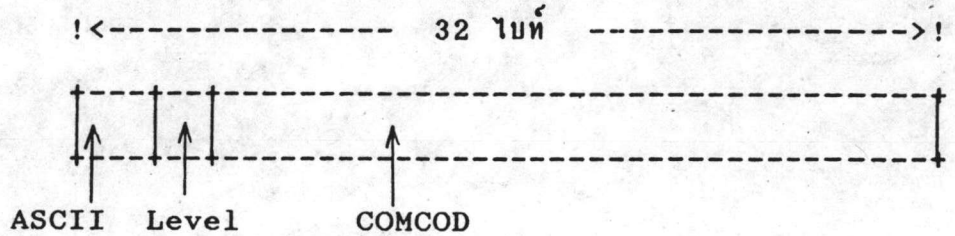
ตารางที่ 3.1 จากรหัส "A" ถึงรหัส "*" ตามลำดับ แต่ละช่องรหัสจะใช้เนื้อที่ 4 ไบท์ โดย 2 ไบท์แรกแทนค่าตำแหน่งของระเบียนของรหัสตัวถัดไป และ 2 ไบท์หลัง แทนค่าตำแหน่งของระเบียนในแฟ้มข้อมูลอักขระที่เก็บรายละเอียดของรหัสนั้น (รูป 3.8)



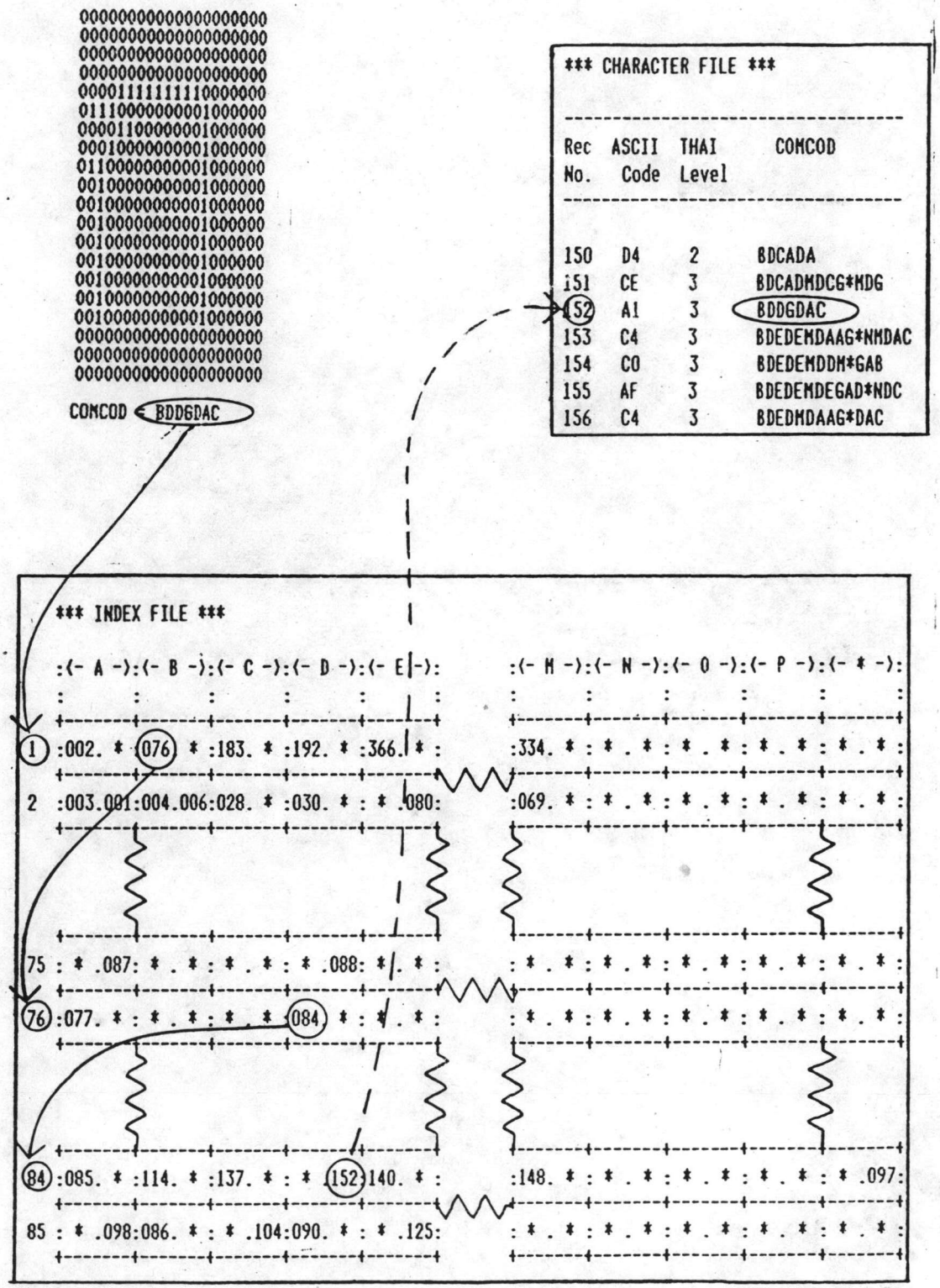
รูปที่ 3.8 แสดงการเก็บข้อมูลในแต่ละระเบียนของแฟ้มข้อมูลดัชนี

3.5.2.2 แฟ้มข้อมูลอักขระ (Character file)

เป็นแฟ้มข้อมูลที่เก็บรายละเอียดของรหัสที่ค้นหานั้นเช่น รหัสแอสกี้ (ASCII Code) ระดับที่ในคำภาษาไทย และรหัสรวมของอักขระแต่ละรูป รายละเอียดของแต่ละระเบียนในแฟ้มข้อมูลอักขระได้แสดงไว้ในรูปที่ 3.9 โดยที่ 2 ไบท์แรกเก็บรหัสแอสกี้ของรหัสรวมนั้น ไบท์ถัดมาเก็บระดับที่ในคำภาษาไทย ส่วนอีก 29 ไบท์สุดท้ายเก็บรหัสรวมของอักขระนั้น



รูปที่ 3.9 แสดงการเก็บข้อมูลในแต่ละระเบียนของแฟ้มข้อมูลอักขระ



รูปที่ 3.10 แสดงการค้นหารหัสรวม "BDDGDAC" ของอักขระ "ก"

ตัวอย่างการค้นหารหัสรวม "BDDGDAC" ของอักขระ "ก" (รูปที่ 3.10) มีดังนี้ เริ่มจากการนำรหัสแรกของรหัสรวมในที่นี้คือรหัส "B" ไปค้นหาที่จุดเริ่มต้นในแฟ้มข้อมูลดัชนี (ระเบียนแรก) ในตำแหน่งช่องรหัส "B" ซึ่ง 2ไบท์แรกมีค่าเป็น 76 และ 2 ไบท์หลังมีค่าเป็น 0 แสดงว่าจะต้องนำรหัสค่าที่ 2 ของรหัสรวมคือรหัส "D" ไปค้นหาในช่องรหัส "D" ในระเบียนที่ 76 ของแฟ้มข้อมูลดัชนี ซึ่งในไบท์ที่ 1-2 และ 3-4 มีค่าเป็น 84 และ 0 ตามลำดับ ขั้นต่อไปให้นำรหัสตัวที่ 3 ของรหัสรวม ในที่นี้คือรหัส "D" ไปค้นหาในช่องรหัส "D" ในระเบียนที่ 84 ของแฟ้มข้อมูลดัชนี ซึ่ง 2 ไบท์แรกมีค่าเป็น 0 ส่วน 2 ไบท์ถัดมามีค่าเป็น 152 แสดงว่ารหัสรวม "BDDGDAC" มีรายละเอียดของข้อมูลเก็บอยู่ในระเบียนที่ 152 ของแฟ้มข้อมูลอักขระ เมื่อนำค่ารหัสต้นแบบในระเบียนที่ 152 ของแฟ้มข้อมูลอักขระมาเปรียบเทียบกับรหัสรวมจะพบว่าเป็นรหัสที่ตรงกัน นั้นคือรหัสรวม "BDDGDAC" มีค่ารหัสแอสกี้ เป็น "A1" เป็นอักขระอยู่ในระดับที่ 3 ของคำในภาษาไทย หากรหัสต้นแบบในแฟ้มข้อมูลอักขระมีค่าไม่เท่ากับรหัสรวมก็จะใช้หลักวิชาสถิติเรื่องความน่าจะเป็นมาประยุกต์ใช้ในการเปรียบเทียบรหัสทั้งสอง หากพบว่ารหัสทั้งสองมีค่าที่เหมือนกันมากกว่า 2/3 แล้ว จะคาดหมายว่ารหัสทั้งสองนั้นเป็นรหัสของอักขระรูปเดียวกัน แต่ถ้ารหัสทั้งสองมีความแตกต่างกันมากก็จะต้องค้นหารหัสต้นแบบตัวถัดไปเรื่อยไปจนกว่าจะค้นหาค่าระเบียนที่เก็บรายละเอียดของรหัสรวมนั้นได้ และถ้าไม่มีค่าที่จะชี้ไปในตำแหน่งของระเบียนถัดไปในแฟ้มข้อมูลดัชนีแล้ว แสดงว่ารหัสรวมของอักขระนั้นไม่สามารถจะตรวจรู้ได้

3.5.3 จำนวนรหัสต้นแบบ

การค้นหาข้อมูลจำนวนน้อยย่อมจะค้นหาได้รวดเร็วกว่าการค้นหาข้อมูลจำนวนมาก แต่การค้นหารหัสต้นแบบเพื่อการตรวจรู้อักขระนี้ จำนวนรหัสต้นแบบยิ่งมีมากเท่าไรโอกาสที่จะตรวจรู้อักขระในแบบพิมพ์ต่างๆ ก็จะมีมากเท่านั้น การวิจัยนี้ได้สร้างแบบพิมพ์ภาษาไทยจำนวน 5 แบบพิมพ์ แต่ละแบบพิมพ์

ประกอบด้วยอักขระจำนวน 70 รูป และแบบพิมพ์ภาษาอังกฤษอีก 1 แบบพิมพ์มีอักขระจำนวน 64 รูป รวมอักขระต้นแบบทั้งสิ้น 414 รูป แต่สามารถแปลงให้เป็นรหัสต้นแบบจำนวน 390 รหัส (ภาคผนวก ก) สาเหตุที่จำนวนรหัสต้นแบบน้อยกว่าอักขระต้นแบบ เพราะอักขระรูปเดียวกันแต่ต่างแบบพิมพ์กันมีค่ารหัสรวมที่เหมือนกัน

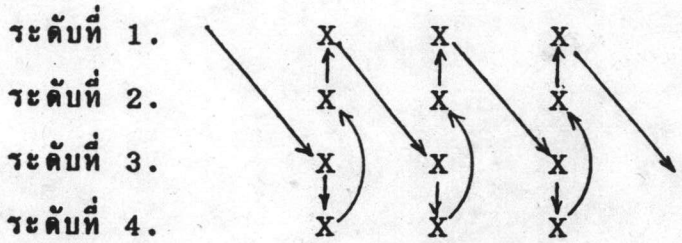
3.5.4 ภาษาคอมพิวเตอร์ที่ใช้เขียนชุดคำสั่ง

ชุดคำสั่งที่ใช้ในงานวิจัยนี้จะเขียนด้วยภาษาเบสิก (Basic Language) เป็นหลัก แต่จะใช้ภาษาแอสเซมบลี้ (Assembly Language) บางส่วนสำหรับลักษณะงานที่ภาษาเบสิกไม่สามารถประมวลผลได้ เช่น การสับเปลี่ยนข้อมูลในหน่วยความจำหลัก (Memory Transfers) แล้วจึงเชื่อมโยงคำสั่งทั้งสองภาษาเข้าด้วยกัน

3.6 ผลลัพธ์ที่ได้จากการตรวจรู้อักขระ (7)

ผลลัพธ์ที่ได้จากการตรวจรู้อักขระจะต้องเป็นรหัสที่เครื่องคอมพิวเตอร์สามารถนำไปประมวลผลได้ เพราะการนำรหัสที่ได้มาประมวลผลด้วยเครื่องคอมพิวเตอร์เป็นการใช้ประโยชน์ของระบบการตรวจรู้อักขระที่มีประสิทธิภาพที่สุด รหัสที่เครื่องคอมพิวเตอร์สามารถจะนำไปประมวลผลได้เช่น รหัสแอสกี้ รหัสเอบซิดิกค์ (EBCDIC Code) รหัสบีซีดี (BCD) เป็นต้น

รูปแบบการจัดคำภาษาไทยจะจัดเรียงอักขระใน 4 ระดับ ดังได้กล่าวมาแล้วในหัวข้อ 3.1 แต่ผลลัพธ์ของการตรวจรู้อักขระภาษาไทยจะเก็บเป็นแถวในระดับเดียวกันหมด เพื่อประหยัดเนื้อที่ที่จะใช้เก็บผลลัพธ์ของการตรวจรู้อักขระภาษาไทยนี้ การรวมอักขระภาษาไทยทั้ง 4 ระดับให้อยู่ในระดับเดียวกันจะจัดเรียงอักขระในระดับที่ 3,4,2,1 ตามลำดับ ถ้าหากอักขระในระดับใดไม่มีก็จะข้ามตำแหน่งของอักขระนั้นไป (รูปที่ 3.11 - 3.12)



รูปที่ 3.11 ลำดับของการจัดเรียงอักขระภาษาไทยในแต่ละระดับให้อยู่ในระดับเดียวกัน

ระดับที่	3	3	3	4	1	3	3	2	1	3
อักขระ	ค	น	ม	ุ	่	ง	ม	ั	่	น

รูปที่ 3.12 แสดงการจัดเรียงอักขระทั้ง 4 ระดับ ของคำว่า "คนมุ่งมั่น" ให้อยู่ในระดับเดียวกัน