

บทที่ 2

รูปแบบการแทนข้อมูล ในเครื่องคอมพิวเตอร์

วัตถุประสงค์

หลังจากเรียนจบบทที่ 2 แล้ว นักศึกษาต้องสามารถ:

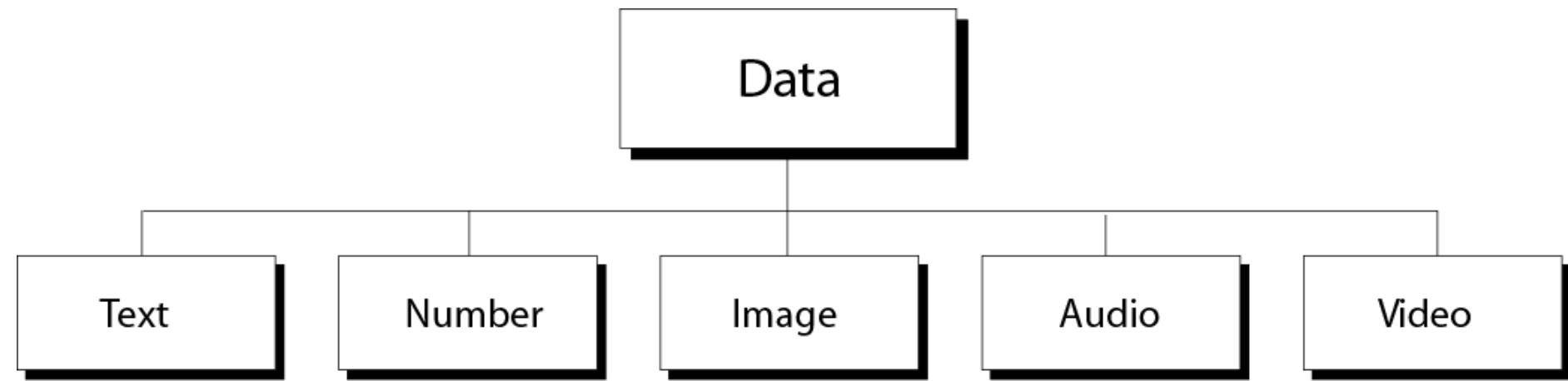
- นิยามความหมายของ data type
- จินตนาการมองเห็นภาพว่าข้อมูลถูกจัดเก็บในคอมพิวเตอร์อย่างไร
- บอกความแตกต่างของข้อมูลประเภท text, numbers, images, video, and audio
- กระทำการทางคณิตศาสตร์กับเลขฐาน 16 และเลขฐาน 8 ได้

2.1

DATA TYPES

รูปที่ 2-1

ประเภทต่างๆของข้อมูล



การประยุกต์คอมพิวเตอร์กับข้อมูลประเภทต่างๆ

- โปรแกรมด้าน **วิศวกรรมศาสตร์** ใช้คอมพิวเตอร์ประมวลผลเกี่ยวกับตัวเลข (number) เป็นส่วนใหญ่ เช่น บวก ลบ คูณ หาร แก้สมการพีชคณิตและสมการตรีโกณมิติ และหารากสมการอนุพันธ์เป็นต้น
- โปรแกรม **ประมวลผลคำ** ใช้คอมพิวเตอร์ประมวลผลข้อความ (text) เช่น เพิ่มข้อความ แทรกคำ ลบข้อความ ย้ายข้อความ เป็นต้น
- โปรแกรม **ประมวลผลภาพ** ใช้คอมพิวเตอร์เพื่อประมวลผลข้อมูลภาพ (image) เช่น สร้างภาพ ย่อภาพ ขยายภาพ หมุนภาพ เป็นต้น
- โปรแกรม **ประมวลผลข้อมูลเสียง** (audio) เช่น ประมวลผลเสียงดนตรี ประมวลเสียงพูดของมนุษย์ บันทึกและค้นหาเสียงของนักร้องเป็นต้น
- โปรแกรม **ประมวลผลภาพแบบวิดีโอ** (video) เช่น การแสดงผลภาพและเสียง การสร้างภาพยนตร์ การสร้าง effects เป็นหลักการสร้างภาพยนตร์เป็นต้น



Note:

The computer industry uses the term “multimedia” to define information that contains numbers, text, images, audio, and video.

2.2

ข้อมูลที่จัดเก็บอยู่ในเครื่องคอมพิวเตอร์

การจัดเก็บข้อมูลในรูป Bit Pattern

- คำถามคือ คอมพิวเตอร์จัดการกับข้อมูลประเภทต่างๆนี้อย่างไร? จำเป็นต้องมี คอมพิวเตอร์หลายประเภทเพื่อดำเนินการกับข้อมูลที่ต่างประเภทกันหรือไม่?
คำตอบคือไม่! คอมพิวเตอร์เครื่องเดียวสามารถประมวลผลข้อมูลประเภทต่างๆ ได้อย่างมีประสิทธิภาพ
- วิธีการคือคอมพิวเตอร์เก็บข้อมูลทุกประเภทโดยใช้รูปแบบที่เป็น **uniform representation** นั่นคือข้อมูลภายใต้รูปแบบเดียวกันของทุกประเภทถูกแปลงให้อยู่ในรูปแบบนี้เมื่อนำไปเก็บในคอมพิวเตอร์ และเมื่อนำออกจากคอมพิวเตอร์ก็จะถูกแปลงกลับเป็นรูปแบบเดิม รูปแบบสากลที่ใช้กันนี้เรียกว่า **Bit Pattern**
- Bit** ย่อมาจาก **binary digit** คือหน่วยที่เล็กที่สุดของข้อมูลที่จัดเก็บในเครื่อง คอมพิวเตอร์ซึ่งอาจเป็น 0 หรือ 1 เป็นการแสดงสถานะของอุปกรณ์ที่มี 2 สถานะ เช่น สวิทช์ไฟฟ้าจะมี 2 สถานะคือ **เปิด (1)** กับ **ปิด (0)** คอมพิวเตอร์ใช้กลุ่มของบิต เพื่อแทนข้อมูล กลุ่มของบิตนี้แหลกที่เรียกว่า **Bit Pattern**

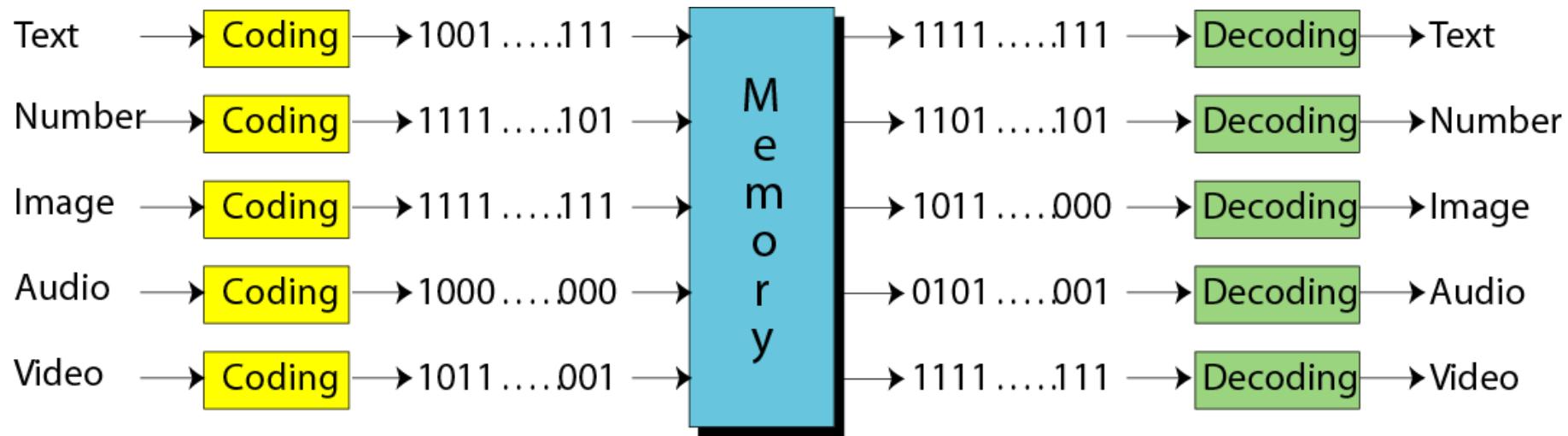
ตัวอย่าง bit pattern จำนวน 16 bit

1 0 0 0 1 0 1 0 1 0 1 1 1 1 1 1

Bit Pattern

- คำถามต่อมาคือ... คอมพิวเตอร์จะรู้ได้อย่างไรว่าข้อมูลกลุ่มใดเป็นข้อมูลประเภทใด? คำตอบคือ **คอมพิวเตอร์ไม่รู้!** หน่วยความจำของคอมพิวเตอร์ทำหน้าที่เฉพาะเก็บข้อมูลในรูป **bit pattern** เท่านั้น เป็นหน้าที่ของ **โปรแกรม** หรือ **อุปกรณ์** ที่นำเข้า/แสดงผล ที่จะต้องตีความหมายเองว่าสิ่งที่เก็บเป็นข้อมูลประเภทใด
- ข้อมูลจะถูก **แปลงรหัส** (code) เมื่อนำเข้าไปเก็บในคอมพิวเตอร์ และจะถูก **แปลงกลับ** (encode) เมื่อนำออกสู่โลกภายนอก
- ตามข้อตกลงทั่วไป bit pattern ที่มีจำนวนบิตเท่ากับ 8 บิตจะเรียกว่า **ไบท์** (byte) คำว่า **ไบท์** สามารถใช้เป็นหน่วยวัดขนาดของหน่วยความจำ และหน่วยความจำสำรองก็ได้

ตัวอย่างของ bit patterns



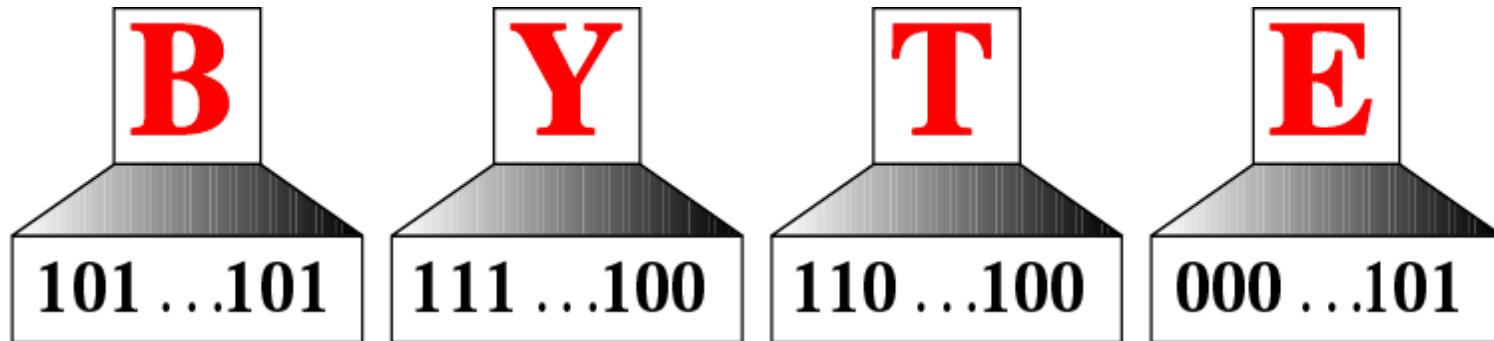
2.3

การแทนข้อมูล

1. การแทนข้อมูลในหน่วยความจำ : Text

- Text ในภาษาไดก์ตานหมายถึงสตริงของอักษรที่แทนความหมายตามภาษาอังกฤษ เช่นในภาษาอังกฤษใช้อักษร 26 ตัว (A,B,C,...) แทนอักษรตัวใหญ่ อักษรเล็ก 26 ตัว (a,b,c,...) แทนตัวอักษรตัวเดิม ตัวอักษร 9 ตัว (0,1,2,3,...) แทนอักษรที่เป็นตัวเลขที่ใช้คำนวณไม่ได้ สัญลักษณ์ที่ใช้แทนเครื่องหมายวรรคตอน เช่น ., ;, :, ?, ! นอกจากนี้ยังมีอักษรที่แทนการขึ้นบรรทัดใหม่ (new line) แทนช่องว่าง (blank) และแทน tab เพื่อการย่อหน้าเป็นต้น
- เราอาจแทนสัญลักษณ์แต่ละตัวด้วย bit pattern เช่นคำว่า BYTE ที่ประกอบด้วยอักษร 4 ตัว แต่ละตัวแทนด้วยกลุ่มของบิตดังนี้

การแทนสัญลักษณ์โดยใช้ bit patterns



การแทนข้อมูลในหน่วยความจำ : Text (ต่อ)

- คำาถามคือเราต้องใช้ **จำนวนกี่บิต** ในการแทนตัวอักษร 1 ตัว? คำาตอบคือ เราต้องดูว่าในภาษาานนๆ มีอักษรทั้งหมดกี่ตัว? ถ้าจำนวนอักษรมีมาก จำนวนบิตที่แทนใน **bit pattern** ก็จะยาวมากขึ้นด้วยเช่น

ถ้าเรามีอักษร 2 ตัว เราจะใช้จำนวนบิตเท่ากับ 1 บิต

คือ 0 กับ 1

ถ้าเรามีอักษร 4 ตัว เราจะใช้จำนวนบิตเท่ากับ 2 บิต

คือ 00 01 10 11

ถ้าเรามีอักษร 8 ตัว เราจะใช้จำนวนบิตเท่ากับ 3 บิต

คือ 000 001 010 011 100 101 110 111

ตารางที่ 2.1 จำนวนสัญลักษณ์และความยาวของ bit pattern

จำนวนสัญลักษณ์

2

4

8

16

...

128

256

...

65,536

ความยาวของ bit pattern

1

2

3

4

...

7

8

...

16



ประเภทของรหัส (Codes)

- กลุ่มของ bit pattern แต่ละแบบแทนกลุ่มของอักษรที่แตกต่างกัน แต่ละกลุ่มเรียกว่า **รหัส (code)** และกระบวนการแทนสัญลักษณ์ด้วยรหัสนี้เรียกว่า **กระบวนการเข้ารหัส (coding)**
 - * รหัส **ASCII**: American National Standard Institute (ANSI) ได้พัฒนารหัสแทนอักษรโดยตั้งชื่อรหัสว่า **American Standard Code for Information Interchange (ASCII)** รหัส ASCII ใช้ 7 บิตแทนอักษรแต่ละตัว หมายความว่า รหัส ASCII สามารถใช้แทนอักษรได้ทั้งหมด $2^7 = 128$ ตัวอักษร

คุณสมบัติเบื้องต้นของรหัส ASCII

- รหัส ASCII ใช้ 7 บิตแทนตัวอักษรเริ่มจาก **0000000** ถึง **1111111**
- Pattern แรกคือ **0000000** แทน **null character**
- Pattern สุดท้ายคือ **1111111** แทน **delete character**
- ในจำนวนรหัสทั้งหมด มี **31** ตัวที่แทน **อักษรควบคุม (control character)**
- ตัวอักษรที่เป็นตัวเลข **0..9** ถูกเข้ารหัสก่อนรหัสตัวอักษร
- รหัสอักษรตัวใหญ่ (A..Z) ถูกเข้ารหัสก่อนอักษรตัวเล็ก (a..z)
- รหัสที่แทนอักษรตัวใหญ่แตกต่างจากรหัสที่แทนอักษรตัวเล็กเพียง **1** บิต เช่น bit pattern แทน A = **1000001** ส่วนของ a = **1100001**

ประเภทของรหัส (Codes)

* รหัส **Extended ASCII**: เพื่อที่จะทำให้แต่ละ bit pattern มีขนาด 1 byte (8 บิต) รหัส ASCII จึงได้เพิ่มบิตขึ้นอีก 1 บิตคือเพิ่มบิต 0 ไว้เป็นบิตแรกใน bit pattern ทำให้ bit pattern มีขนาดพอดีกับ 1 ไบท์ในหน่วยความจำ นั่นคือสำหรับรหัส Extended ASCII จะมีรหัสตั้งแต่ **00000000** ถึง **01111111**

ข้อสังเกต มีผู้ผลิตหารดแวร์บางรายพยายามที่จะใช้บิตที่เพิ่มขึ้นเพื่อสร้างอักษร หรือสัญลักษณ์ตัวใหม่เพิ่มเติม แต่ก็ยังไม่ประสบความสำเร็จ เพราะแต่ละรายยังใช้ มาตรฐานที่แตกต่างกัน

* รหัส **EBCDIC**: ในการพัฒนาคอมพิวเตอร์ระยะแรกๆ บริษัทไอบีเอ็มจำกัด ได้ พัฒนารหัสที่เรียกว่า **Extended Binary Coded Decimal Interchange Code** รหัสนี้ใช้ bit pattern จำนวน 8 บิตซึ่งสามารถแทนตัวอักษรได้ถึง $2^8 = 256$ ตัวอักษร อย่างไรก็ตาม รหัส EBCDIC ก็ไม่ได้ใช้ในเครื่องคอมพิวเตอร์อื่นๆ นอกจากเครื่อง **IBM mainframe** เท่านั้น

ประเภทของรหัส (Codes)

*รหัส **Unicode**: รหัสที่ก่อตัวมาในตอนต้นทั้งหมดเป็นรหัสที่คิดขึ้นเพื่อแทนอักษรภาษาอังกฤษเท่านั้น ด้วยเหตุนี้จึงจำเป็นที่จะต้องมีรหัสที่สามารถแทนอักษรภาษาอื่นๆที่มิอยู่ในโลกนี้ด้วย เป็นเหตุให้ผู้ผลิตหาร์ดแวร์และผู้ผลิตซอฟท์แวร์ได้ร่วมมือกันออกแบบรหัสชนิดใหม่และให้ชื่อว่า **Unicode**

ใช้ **16 บิตแทน 1 bit pattern** ซึ่งสามารถแทนสัญลักษณ์ได้ **65,536** (2^{16}) ตัว แต่ละตัวของรหัสสามารถใช้แทนอักษรที่แตกต่างกันของภาษาในโลกนี้ บางตัวของ bit pattern ก็ใช้สำหรับอักษรที่แทนกราฟฟิกและอักษรพิเศษ

ภาษา JAVA ใช้ Unicode แทนอักษรที่ใช้ในการเขียน

MicroSoft Windows ใช้เพียง 256 ตัวแรกของรหัส

* **ISO**: องค์กรของสหราชูปถัมภ์ International Organization for Standardization มีชื่ออยู่ว่า ISO ได้ออกแบบรหัสโดยใช้ 32 บิต ซึ่งสามารถแทนอักษรได้ **4,294,967,296** (232)

ประเภทของรหัส (Codes)

#ภาษา JAVA ใช้ Unicode แทนอักษรที่ใช้ในการเขียน

#MicroSoft Windows ใช้เพียง 256 ตัวแรกของรหัส

*รหัส ISO: องค์กรของสหราชอาณาจักรชื่อ International

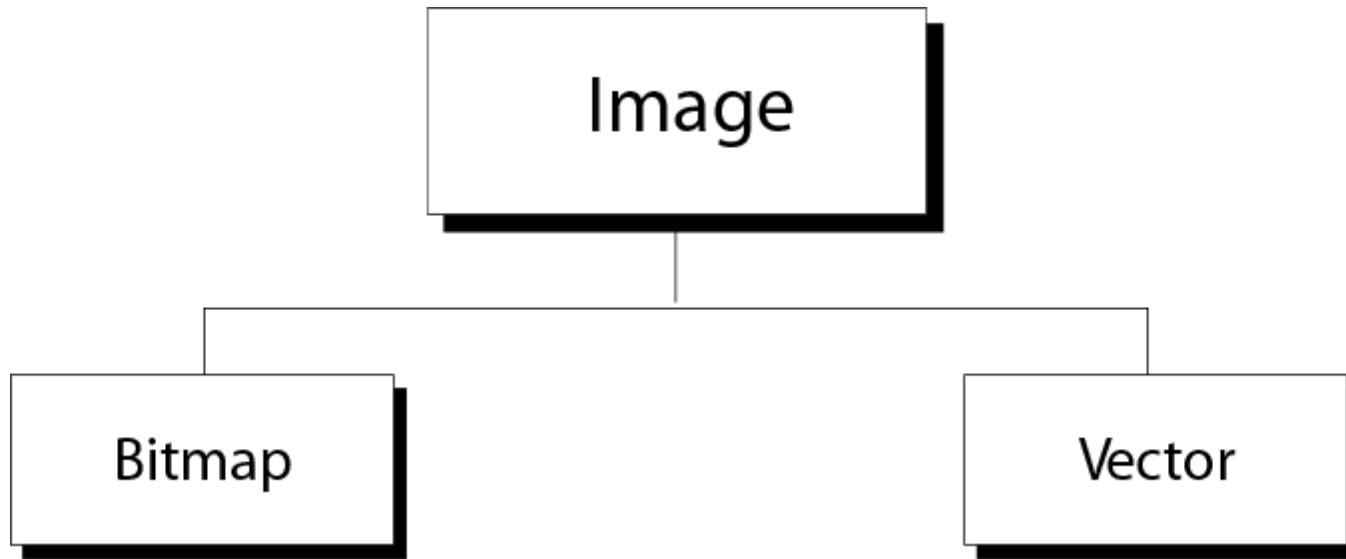
Organization for Standardization มีชื่อย่อว่า ISO ได้
ออกแบบรหัสโดยใช้ 32 บิต ซึ่งสามารถแทนอักษรได้
4,294,967,296 (2^{32}) ตัว

การแทนคำว่า “BYTE” ในรูปรหัส ASCII

B	Y	T	E
1000010	1011001	1010100	1000101

รูปที่ 2-6

การแทนข้อมูลในหน่วยความจำ : **Image**

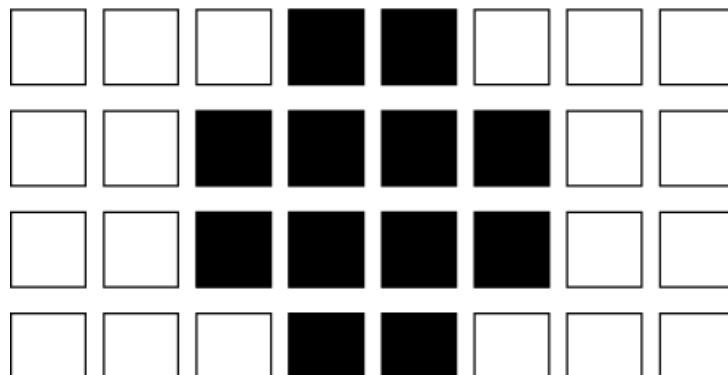


2. การแทนข้อมูลในหน่วยความจำ : **Image**

Bitmap Graphic: วิธีนี้ image จะถูกแบ่งเป็นแมตริกส์ของ pixel โดยแต่ละ pixel เป็นจุดเล็กๆ ซึ่งขนาดของ pixel ขึ้นอยู่กับ resolution เช่นภาพอาจแบ่งออกเป็น 1,000 pixels หรือ 10,000 pixels เมื่อ resolution ของแบบที่สองจะละเอียดกว่าแบบแรกแต่ก็ใช้หน่วยความจำมากกว่าในการจัดเก็บ

หลังจากแบ่งเป็น pixel แล้วแต่ละ pixel จะแทนด้วย bit pattern ขนาดและค่าของ pattern ขึ้นอยู่กับประเภทของภาพ เช่นถ้าเป็นภาพขาวดำ 1 pixel ใช้ 1-bit pattern แทนก็เพียงพอ โดย pattern 1 แทนจุดดำ (black pixel) และ pattern 0 แทนจุดขาว (white pixel) จากนั้น pattern จะถูกจัดเก็บใน pixel ต่อ pixel ตามลำดับ (ดังรูป 2-7)

วิธี bitmap graphic ของรูปขาวดำ



Image

00011000

00111100

00111100

00011000

Matrix Representation

00011000 00111100 00111100 00011000

Linear Representation

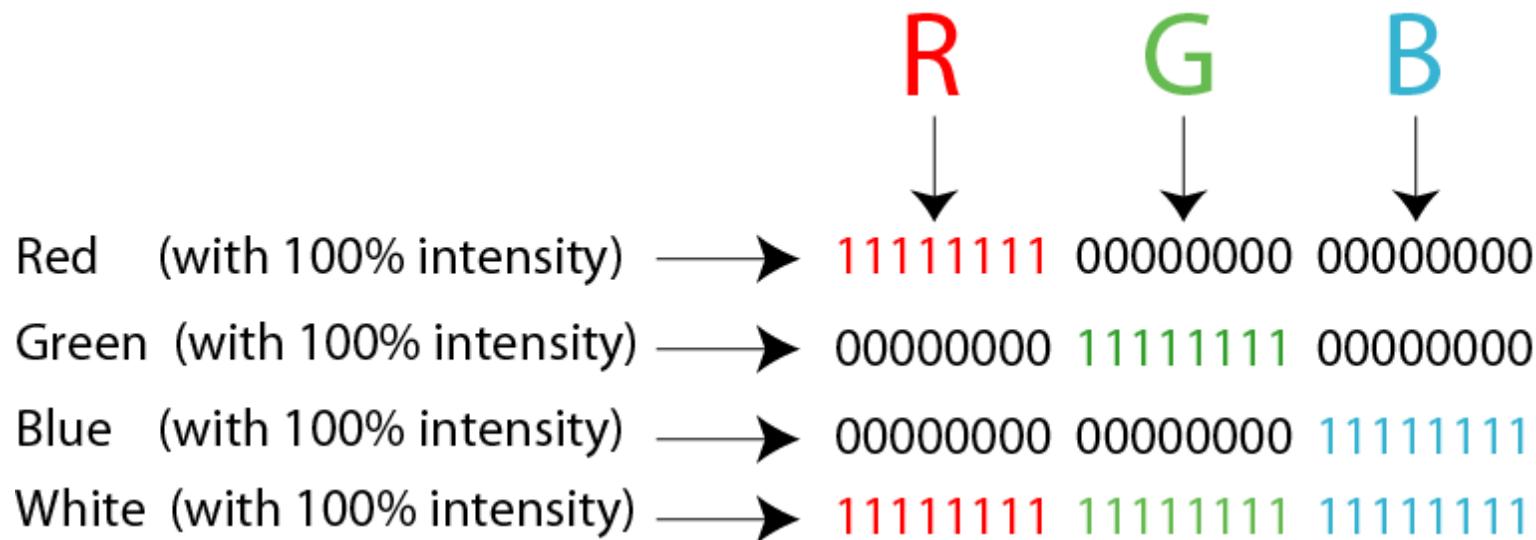
การแทนข้อมูลประเภท Color Image

- ในกรณีที่ข้อมูลภาพเป็นสี เราจะต้องเพิ่มขนาดของ bit pattern ให้ยาวขึ้นเพื่อแทน gray scale เช่น ถ้าเราต้องการแสดง gray level 4 ระดับ เราสามารถใช้ 2-bit pattern โดย pixel สีดำจะแทนโดย 11 pixel สี dark gray แทนด้วย 10 pixel สี light gray แทนด้วย 01 และ pixel สีขาวแทนด้วย 00
- การแทนข้อมูลภาพสี แต่ละจุดสีจะแยกออกเป็นสีพื้นฐาน 3 สีคือ สีแดง (Red) สีเขียว (Green) และสีน้ำเงิน (Blue) หรือย่อเป็น RGB จากนั้นก็จะทำการวัดความเข้ม (intensity) ของแต่ละสีในแต่ละ pixel แล้วจึงกำหนด bit pattern ให้กับ pixel (ปกติกำหนดเป็น 8)

การแทนข้อมูลประเภท Color Image

- กล่าวอีกอย่างหนึ่งคือแต่ละ pixel มี **3-bit pattern** โดย bit pattern แรกแทนความเข้มของสีแดง bit pattern ที่สองแทนความเข้มของสีเขียว และ bit pattern ที่สามแทนความเข้มของสีน้ำเงิน ดังตัวอย่างในภาพต่อไปนี้

การแทน color pixels



การแทนข้อมูลในหน่วยความจำ : **Image**

Vector Graphic: ปัญหาของการแทนข้อมูลภาพด้วยวิธี bitmap คือจำนวน bit pattern ที่แน่นอนที่แทนข้อมูลภาพจะถูกจัดเก็บในเครื่องคอมพิวเตอร์ ถ้าเราต้องการย่อหรือขยายภาพ เราจะต้องเปลี่ยนขนาดของ pixel ซึ่งจะทำให้ภาพดูเป็นรอยย่นหรือขยายเป็นเม็ดๆ

สำหรับวิธี **Vector Graphic** จะไม่เก็บเป็น bit pattern แต่จะแบ่งภาพออกเป็น curves และ lines แต่ละ curve และ line จะแทนด้วยสมการทางคณิตศาสตร์ เช่น line อาจแทนด้วยพิกัดของจุดเริ่มต้นและจุดสุดท้าย วงกลมอาจแทนด้วยพิกัดของจุดศูนย์กลางกับความยาวของรัศมี เป็นต้น

การแทนข้อมูลในหน่วยความจำ : **Image**

- กลุ่มของสูตรทางคณิตศาสตร์ที่แทนข้อมูลภาพจะถูกจัดเก็บในคอมพิวเตอร์ เมื่อต้องการแสดงหรือพิมพ์ภาพ เราต้องกำหนดของภาพให้ชัดเจนโดยกำหนดเป็น **input** จากนั้นระบบจะทำการคำนวณสูตรของภาพโดยใช้ข้อมูลที่กำหนดแล้วจึงพิมพ์หรือแสดงภาพนั้นออกมาทางหน้าจอ จะเห็นว่าทุกครั้งที่เราต้องการพิมพ์หรือแสดงภาพทางหน้าจอ สูตรที่เก็บภาพจะถูกคำนวณใหม่ทุกครั้ง
- ให้นักศึกษาเปรียบเทียบจุดอ่อนและจุดแข็งของวิธีการจัดเก็บข้อมูลภาพทั้งสองแบบ แล้วสรุปเองว่าวิธีใดเหมาะสมกับการประยุกต์ใช้งานในลักษณะใด

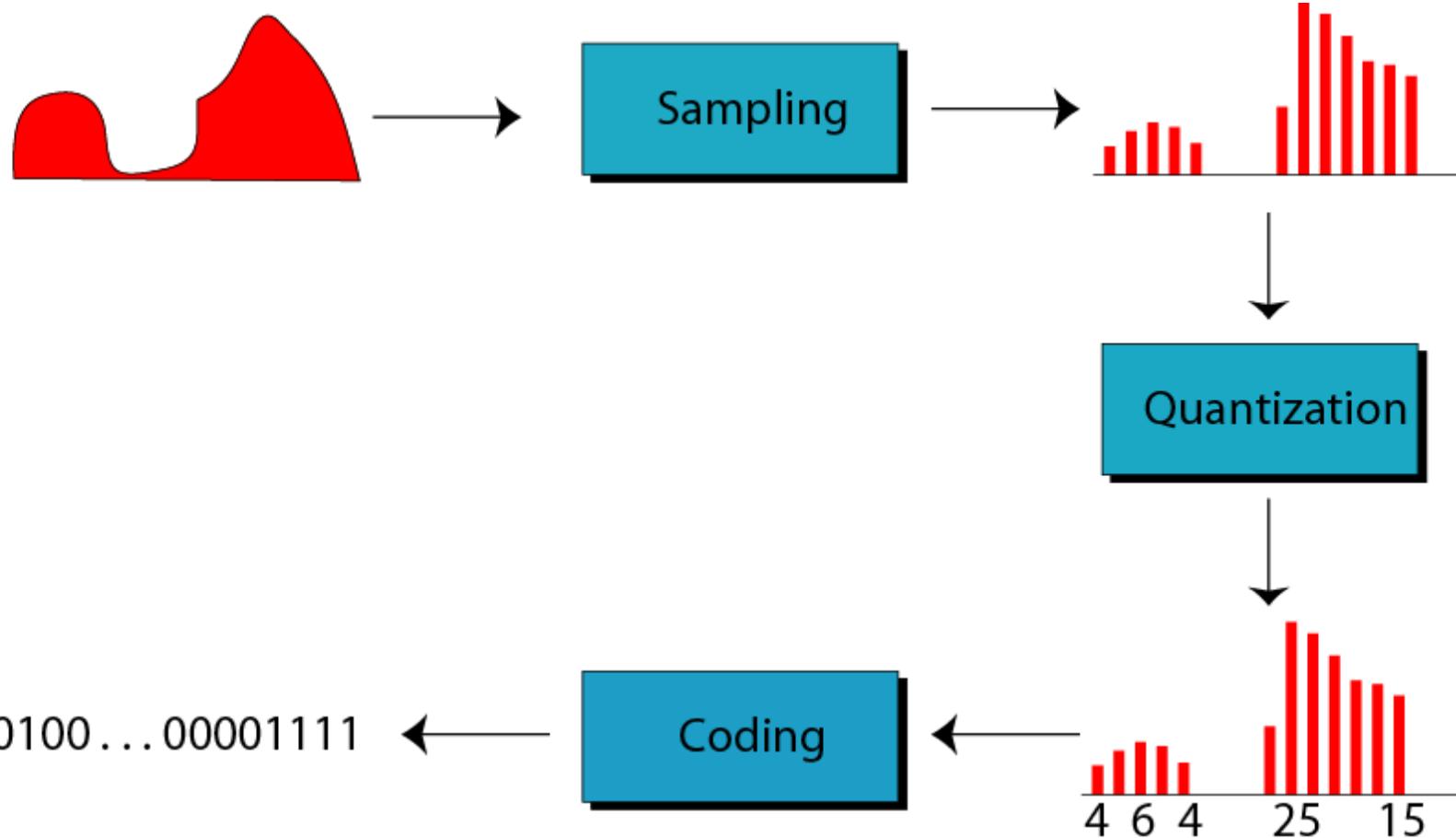
3. การแทนข้อมูลในหน่วยความจำ : Audio

- ข้อมูลประเภท **audio** ได้แก่ ข้อมูลที่เป็นเสียง (sound) และดนตรี (music) แนวคิดคือการแปลง **ข้อมูล audio** ไปเป็น **ข้อมูลดิจิตอล** และใช้ **bit pattern** เก็บข้อมูลเหล่านี้ โดยธรรมชาติแล้ว ข้อมูล audio เป็นข้อมูล **analog** ที่มีลักษณะเป็นคลื่นที่มีความต่อเนื่อง (continuous) ไม่เหมือนกับข้อมูลประเภทจำนวนเต็มที่แทนจำนวนนักศึกษาในแต่ละชั้น ปี จำนวนผู้มีสิทธิลงคะแนนเสียงในแต่ละจังหวัด หรือจำนวนลูกค้าที่เข้ามาซื้อสินค้าในห้างสรรพสินค้าแห่งหนึ่งในแต่ละวัน ข้อมูลประเภทนี้ เป็นข้อมูล **discrete** (digital)

การแปลงข้อมูล audio เป็น bit pattern

1. ทำการสุ่มสัญญาณ analog ที่แทนเสียง การสุ่มหมายถึงการวัดค่าความสูงของสัญญาณ ณ ช่วงที่เท่ากัน
2. ทำการ quantize ค่าที่ได้จากการสุ่มในขั้นที่ 1 กระบวนการ quantization หมายถึงการกำหนดค่าจากเซตหนึ่งให้กับค่าที่สุ่มมาได้ตัวอย่างเช่น ถ้าค่าที่สุ่มมาได้เท่ากับ 29.2 และเซตที่กำหนดให้เป็นเซตของจำนวนเต็มที่มีค่าอยู่ระหว่าง 0 ถึง 63 เราอาจ quantize ค่าเป็น 29
3. ทำการเปลี่ยนค่า quantize ที่ได้ให้เป็น bit pattern เช่นถ้าค่า quantize ที่ได้มีค่า 25 ค่า bit pattern จะเป็น 00011001 เป็นต้น
4. นำ binary pattern ที่ได้จากขั้นตอนที่ 3 ไปจัดเก็บในคอมพิวเตอร์

การแทนข้อมูลเสียง



4. การแทนข้อมูลในหน่วยความจำ : Video

- **Video** เป็นการแทน **ข้อมูล image** (เรียกว่า frame) ตามเวลา เช่น ภาพยนตร์คือลำดับของเฟรมที่แสดงต่อเนื่องกันเฟรมต่อเฟรมเพื่อสร้างภาพการเคลื่อนไหว ดังนั้นถ้าเราใช้วิธีการจัดเก็บข้อมูลภาพทั่วไปในคอมพิวเตอร์ เราจะใช้วิธีการจัดเก็บข้อมูล video โดยไม่ยาก นั่นคือแต่เฟรมจะถูกเปลี่ยนเป็น bit pattern และทำการเก็บ bit pattern เหล่านี้
- **ข้อสังเกต:** โดยปกติในปัจจุบันข้อมูลประเภท video จะถูกบีบอัด (compress) เพื่อให้ใช้เนื้อที่ในการจัดเก็บน้อยลง วิธีการบีบอัดที่เราได้ยินอยู่ เช่น MPEG จะอธิบายละเอียดในบทที่ 15

2.4

ເລບສັນ 16

เลขฐาน 16

- **Bit pattern** ออกแบบมาเพื่อแทนข้อมูลเมื่อเก็บในคอมพิวเตอร์ แต่จะพบว่าการทำ bit pattern เป็นเรื่องยากสำหรับมนุษย์ การที่จะต้องเขียน 0 และ 1 เป็นແຕวຍາວๆ เป็นเรื่องที่น่าเบื่อและอาจก่อให้เกิดความผิดพลาดได้ง่าย การใช้เลขฐาน 16 จะช่วยแก้ปัญหานี้
- **Hexadecimal notation** เป็นการแทนตัวเลขโดยใช้ฐาน 16 (*hexadec* เป็นภาษากรีก หมายถึง 16) หมายความว่ามีสัญลักษณ์ที่ใช้ 16 ตัวคือตัวเลข 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, และ F
- ประโยชน์ของเลขฐาน 16 จะเห็นชัดเจนเมื่อเราแปลง bit pattern ที่ใช้แทนข้อมูลไปเป็นเลขฐาน 16



Note:

*A 4-bit pattern can be represented
by a hexadecimal digit,
and vice versa.*

ตารางที่ 2.2 Hexadecimal digits

Bit Pattern	Hex Digit	Bit Pattern	Hex Digit
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

การแปลง bit pattern เป็นเลขฐาน 16

- การแปลงจาก bit pattern เป็นเลขฐาน 16 ทำได้โดยการจัดกลุ่มเลข 0 และ 1 ใน bit pattern ออกเป็นกลุ่มๆ ละ 4 บิตแล้วหาค่าเลขฐาน 16 ของแต่ละกลุ่ม เช่น $1111110010010110 = \textcolor{red}{1111} \textcolor{blue}{1100} \textcolor{blue}{1001} \textcolor{teal}{0110} = \textcolor{red}{\text{FC}} \textcolor{blue}{\text{96}}$
- การแปลงจากเลขฐาน 16 เป็น bit pattern ทำโดยการแปลงเลขฐาน 16 แต่ละตัวให้เป็นเลข 0 และ 1 จำนวน 4 บิต(เลขฐาน 2) เช่น
 - ถ้าเลขฐาน 16 มีค่า = **AD37** จะเห็นว่า A = 1010, D = 1101,
 - 3 = 0011 และ 7 = 0111 ดังนั้น AD37 เมื่อแปลงเป็น bit pattern
 - แล้วจะได้ **1010110100110111**

รูปที่ 2-10

การแปลงเลขฐาน 2 เป็นเลขฐาน 16

1 1 1 1	1 1 0 0	1 1 1 0	0 1 0 0	
F	C	E	4	

Hexadecimal



ตัวอย่างที่ 1

จงแปลงเลขฐาน 2: 1100 1110 0010 ให้เป็นเลขฐาน 16

วิธีทำ

แปลงแต่กกลุ่มที่มี 4 บิตเป็นเลขฐาน 16 หนึ่งตัวจะได้เป็น

CE2

ตัวอย่างที่ 2

จงแปลง 0011100010 ให้เป็นเลขฐาน 16

วิธีทำ

แบ่ง bit pattern ออกเป็นกลุ่มของ 4-bit (จากทางขวาմօ).
ในการนี้จะต้องเติมเลข 0 อีก 2 ตัวทางด้านซ้าย จะได้เป็น
000011100010 ซึ่งเปลี่ยนเป็นเลขฐาน 16 ได้เป็น 0E2

ตัวอย่างที่ 3

จงหา bit pattern ของเลขฐาน 16: 24C

วิธีทำ

แปลงเลขเต็ตตัวเป็น bit pattern จะได้: **001001001100.**

2.5

ເລີບຈົ້ານ 8

เลขฐานแปด (Octal notation)

- อีกรูปแบบหนึ่งของการจัดกลุ่ม bit pattern คือจัดเป็นเลขฐาน 8
- Octal notation เป็นการแทนเลขโดยใช้ฐาน 8 (oct เป็นภาษากรีก หมายถึง 8) หมายความว่าใช้เลขเพียง 7 ตัวคือ 0, 1, 2, 3, 4, 5, 6, และ 7 ประโยชน์ของเลขฐาน 8 จะเห็นชัดเจนเมื่อมีการเปลี่ยนการแทนข้อมูลจาก bit pattern เป็นเลขฐาน 8 (เช่นเดียวกับเลขฐาน 16)
- เลขแต่ละตัวของฐาน 8 สามารถแทนด้วย 3 บิต และเลขจำนวน 3 บิตสามารถแทนเลขฐาน 8 ได้ 1 ตัว
- A 3-bit pattern can be represented by an octal digit, and vice versa.

ตารางที่ 2.3 เลขฐาน 8

Bit Pattern	Oct Digit	Bit Pattern	Oct Digit
000	0	100	4
001	1	101	5
010	2	110	6
011	3	111	7

รูปที่ 2-11

การแปลงจากเลขฐาน 2 เป็นเลขฐาน 8

1	1	1	1	1	1	0	0	1	1	1	0	0	0	1	0	0
1	7	6	3	4	4											

Octal

ตัวอย่างที่ 4

จงแปลง bit pattern 101110010 ให้เป็นเลขฐาน 8

วิธีทำ

แบ่ง bit pattern ออกเป็นกลุ่มของ 3-บิตแล้วแปลงเป็นเลข

ฐาน 8 จะได้เป็น 562

ตัวอย่างที่ 5

จงหาค่าเลขฐาน 8 ที่มีค่าตรงกับ bit pattern 1100010

วิธีทำ

แบ่ง bit pattern ออกเป็นกลุ่มของ 3-บิต (จากทางขวา) ในกรณีนี้ต้องเพิ่ม 0 เข้าไป 2 ตัวทางด้านซ้ายจะได้เป็น 001100010 ซึ่งเมื่อแปลงเป็นเลขฐาน 8 ได้ 142

ตัวอย่างที่ 6

จงหา bit pattern สำหรับเลขฐาน 8 ที่มีค่าเท่ากับ 24_8

วิธีทำ

แปลงเลขเต็ลละตัวเป็น bit pattern ได้ 010100

คำสำคัญในบทที่ 2

- ANSI
- binary digit
- digital signal
- hexadecimal
- octal notation
- sampling
- vector graphic
- ASCII
- bitmap graphic
- extended ASCII
- image data
- pixel
- unicode
- bit pattern
- EBCDIC
- ISO
- quantization
- video data
- analog signal