컴퓨터 시스템 일반

Sangwook Lee
Deogi High School

I 정보의 표현

1 수의 체계

2 디지털 정보의 연산

3 디지털 정보의 표현

3 디지털 정보의 표현

1. 문자 데이터의 표현
2. 그림 데이터의 표현
3. 소리 데이터의 표현
4. 동영상 데이터의 표현

1. 문자 데이터의 표현

〈 학습 목표 〉

- 컴퓨터 시스템에서 처리하는 문자 데이터의 표현 원리를 설명할 수 있다.

1. 문자 데이터의 표현 (p.29)

한글, 영문자, 특수 문자 등을

컴퓨터 시스템에서 표현(저장)하려면

이들 문자를 구별하기 위해

각각의 문자별로 미리 약속된 0과 1의 조합,

즉 2진수 형태로 바꾸어 주어야 한다

1. 문자 데이터의 표현 (p.29)

- **문자 코드 (character code)**
  - 문자를 구별하여 표현하기 위해 정해 놓은 규칙(약속)
  - 같은 의미로 사용되는 말
    - 문자 세트 (character set)
    - 문자 인코딩 (character encoding)

- **문자 코드에 의해 결정되는 것**
  - 문자 종류 (☞ 어떤 문자들을 사용할 것인가?)
  - 문자 번호 (☞ 각각의 문자에 몇 번을 배정할 것인가?)
  - 문자 저장 방식 (☞ 문자 번호를 어떤 형태로 저장할 것인가?)

몇 비트로 저장? 바이트를 어떤 순서로 저장?

1. 문자 데이터의 표현 (p.29)

많이 사용하는 문자 코드로는

아스키코드(ASCII)와 유니코드(Unicode)가 있음

문자 코드가 다르면

같은 문자라도 다른 2진수 형태로 표현(저장)됨

1. 문자 데이터의 표현 (p.29)

예를 들어, 문자 A를…

아스키코드로 표현하면
$01000001_{(2)}$

유니코드(UTF-16)로 표현하면
$00000000\ 01000001_{(2)}$

1. 문자 데이터의 표현 (p.29)

- 대표적인 문자 코드
  - 아스키(ASCII)
    - American Standard Code for Informationn Interchange
      - ☞ 미국 정보 교환 표준 부호
  - 엡시딕(EBCDIC)
    - Extended Binary Coded Decimal Interchange Code
      - ☞ 확장 이진화 십진법 교환 부호
  - 유니코드(Unicode)
    - UTF-8
    - UTF-16 BE(Big Endian)
    - UTF-16 LE(Little Endian)

1. 문자 데이터의 표현 (p.29)

- 아스키(ASCII)코드
  - 미국 표준 협회가 만든 코드
  - 7비트로 128개의 문자 표현

2진수	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAM	%	5	E	U	e	u
0110	ACK	SYN	^	6	F	V	f	v
0111	BEL	ETB	&	7	G	W	g	w
1000	BS	CAN	'	8	H	X	h	x
1001	HT	EM	(	9	I	Y	i	y
1010	LF	SUB	)	:	J	Z	j	z
1011	VT	ESC	*	;	K	[	k	{
1100	FF	FC	+	<	L		l	:
1101	CR	QS	,	=	M	]	m	}
1110	SO	RS	.	>	N		n	~
1111	SI	US	/	?	O		o	DEL

상위 비트

하위 비트

$100\ 0001_{(2)} = 65$

상위 비트 하위 비트

[아스키코드 문자표]

1. 문자 데이터의 표현 (p.29)

[예제] 아스키코드 문자표를 보고 다음 문자열을 아스키코드로 표현해 보자.

a=7.5

〈풀이〉

a : 110 0001

= : 011 1101

7 : 011 0111

. : 010 1110

5 : 011 0101

따라서

a=7.5를 아스키코드로 표현하면

01100001 00111101 00110111 00101110 00110101

61 3D 37 2E 35

1. 문자 데이터의 표현 (p.30)

- 유니코드(Unicode)
  - 전 세계의 모든 문자를 표현하기 위해 개발된 국제 표준 코드

- 유니코드 인코딩 방식
  - UTF-8
    - 8 / 16 / 24 / 32 비트로 문자를 표현
      (예: 영문 8비트, 한글 24비트)
  - UTF-16 LE (또는 UTF-16 BE)
    - 16 / 32 비트로 문자를 표현
      (예: 영문 16비트, 한글 16비트)

<유니코드 한글 번호>
가 AC00 (44032)
각 AC01 (44033)
:
힢 D7A2 (55202)
힣 D7A3 (55203)

[유니코드 문자표]

1. 문자 데이터의 표현 (p.30)

- 유니코드 한글 11,172자 문자 번호

가 AC00 (44032)
각 AC01 (44033)
갂 AC02 (44034)
갃 AC03 (44035)
간 AC04 (44036)

힟 D79F (55199)
힡 D7A0 (55200)
힢 D7A1 (55201)
힣 D7A2 (55202)
힤 D7A3 (55203)

	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7	AC8	AC9	ACA	ACB	ACC	ACD	ACE	ACF
0	가 AC00	감 AC10	갠 AC20	갰 AC30	걀 AC40	걐 AC50	걠 AC60	거 AC70	검 AC80	겐 AC90	겠 ACA0	결 ACB0	겤 ACC0	곍 ACD0	고 ACE0	곰 ACF0
1	각 AC01	갑 AC11	갡 AC21	갱 AC31	걁 AC41	걑 AC51	걡 AC61	걱 AC71	겁 AC81	겑 AC91	겡 ACA1	겱 ACB1	겥 ACC1	곎 ACD1	곡 ACE1	곱 ACF1
2	갂 AC02	값 AC12	갢 AC22	갲 AC32	걂 AC42	걒 AC52	걢 AC62	걲 AC72	겂 AC82	겒 AC92	겢 ACA2	겲 ACB2	겦 ACC2	곒 ACD2	곢 ACE2	곲 ACF2
3	갃 AC03	갓 AC13	갣 AC23	갳 AC33	걃 AC43	걓 AC53	걣 AC63	걳 AC73	것 AC83	겓 AC93	겣 ACA3	겳 ACB3	겧 ACC3	곓 ACD3	곣 ACE3	곳 ACF3
4	간 AC04	갔 AC14	갤 AC24	갴 AC34	걄 AC44	걔 AC54	걤 AC64	건 AC74	걷 AC84	걜 AC94	걬 ACA4	걵 ACB4	계 ACC4	곔 ACD4	곤 ACE4	곴 ACF4

[유니코드 한글 문자표]

:: 바이트 저장 순서 (byte order) ::

- 바이트(byte)란
  - 정보의 기본 단위
  - 한 문자를 표현할 수 있는 최소 단위로서, 8개의 비트로 구성

- 바이트 저장 순서

  컴퓨터가 저장하는 데이터는 보통 4바이트(32비트)로 구성되는데
  바이트를 메모리에 순서대로 저장하는 방식은 CPU에 의해 결정되며
  다음과 같이 두 가지 방식이 있음

  - 빅 엔디언(big endian)
  - 리틀 엔디언(little endian)

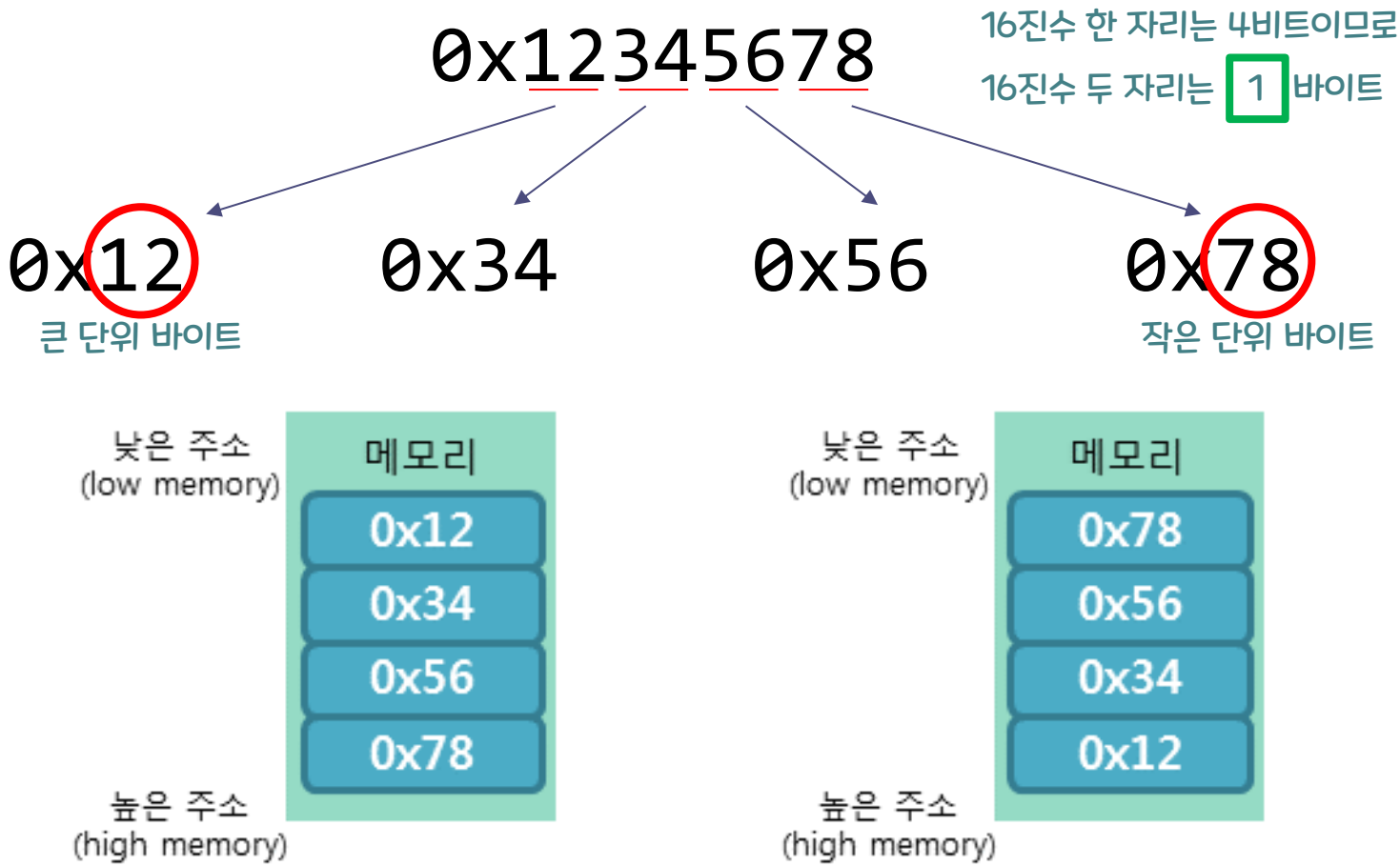
:: 바이트 저장 순서 (byte order) ::

- **빅 엔디언** ☞ 사람이 숫자를 기록하는 방식과 동일
  - 메모리의 앞쪽(낮은 주소)에 큰 단위 바이트를 저장하는 방식
  - 대표적인 CPU: SPARC, Motorola

- **리틀 엔디언**
  - 메모리의 앞쪽(낮은 주소)에 작은 단위 바이트를 저장하는 방식
  - 대표적인 CPU: Intel x86, AMD

:: 바이트 저장 순서 (byte order) ::

예) 0x12345678 이 메모리에 저장된 형태



<빅 엔디언>

<리틀 엔디언>

2 디지털 정보의 연산

1. 문자 데이터의 표현
2. 그림 데이터의 표현
3. 소리 데이터의 표현
4. 동영상 데이터의 표현

2. 그림 데이터의 표현

〈 학습 목표 〉

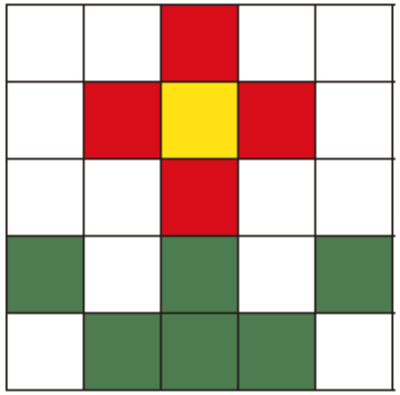
- 컴퓨터 시스템에서 처리하는 그림 데이터의 표현 원리를 설명할 수 있다.

2. 그림 데이터의 표현 (p.31)

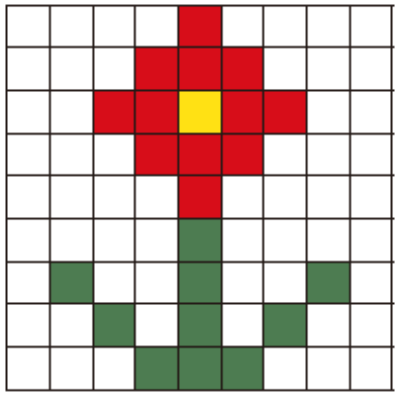
- 픽셀(pixel) = 화소(畫素)
  - Picture Element의 합성어
  - 그림이나 화면을 구성하는 기본 단위
  - 색상 정보를 가지는 최소 단위
    - ☞ 하나의 픽셀이 동시에 여러 색상을 표현할 수 없다는 것을 의미

2. 그림 데이터의 표현 (p.31)

- 해상도(resolution)
  - 1인치당 픽셀 수 (또는 전체의 픽셀 수)
  - 그림이나 화면이 표현할 수 있는 정보량을 결정
  - 해상도가 높을수록 더 정밀(섬세)한 그림의 표현 가능



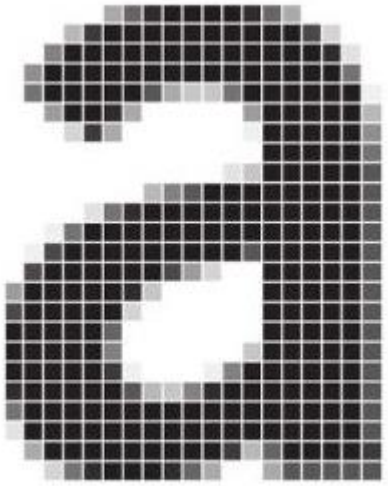
▲ 5픽셀 × 5픽셀 해상도



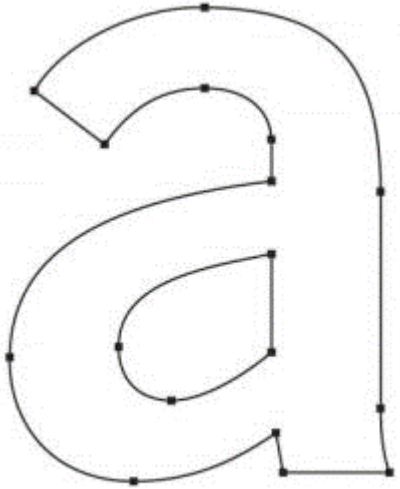
▲ 9픽셀 × 9픽셀 해상도

2. 그림 데이터의 표현 (p.31)

- 그림 정보의 표현 방식 2가지
  - 래스터 방식 (= 비트맵 방식)
  - 벡터 방식



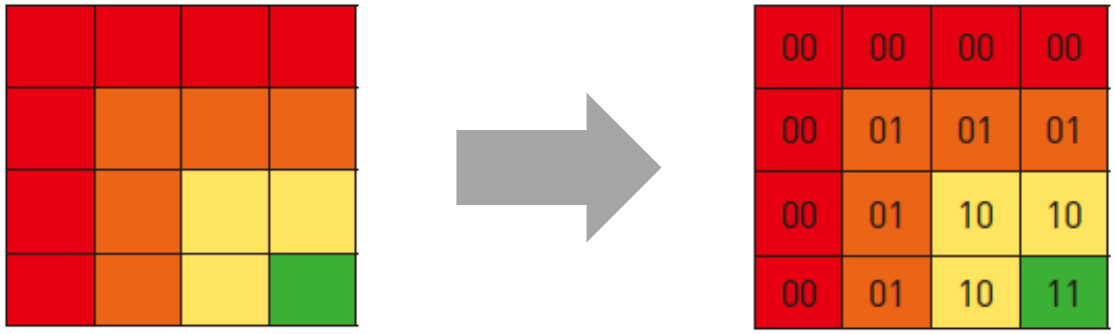
RASTER



VECTOR

2. 그림 데이터의 표현 (p.31)

- 비트맵(bitmap) 방식
  - 픽셀의 색상 정보로 그림을 표현하는 방식
    (색상별로 번호를 정한 후 모든 픽셀의 색상 값(번호)을 저장함)



n개의 비트로 색상을 표현하면, 최대 2^n개의 색상을 표현할 수 있음

(픽셀당 사용하는 비트 수 많을 수록, 표현 가능한 색상 수도 많아짐)

2. 그림 데이터의 표현 (p.31)

(Quiz)

한 픽셀의 색을 표현(저장)할 때 4비트를 사용하는 그림은

최대 16 가지의 색을 표현할 수 있음

우리가 많이 사용하는 JPG나 PNG 파일의 경우

한 픽셀의 색상을 표현하기 위해

Red, Green, Blue 색상 계열(채널)별로 구분하여

각각 8비트의 색상 값으로 표현함

이 경우 한 픽셀을 표현하기 위해 8+8+8=24비트가 사용되며

이러한 그림이 표현할 수 있는 최대 색상수는

$2^8 \times 2^8 \times 2^8 = 2^{24} = 16.7M$(천육백만)

2. 그림 데이터의 표현 (p.31)

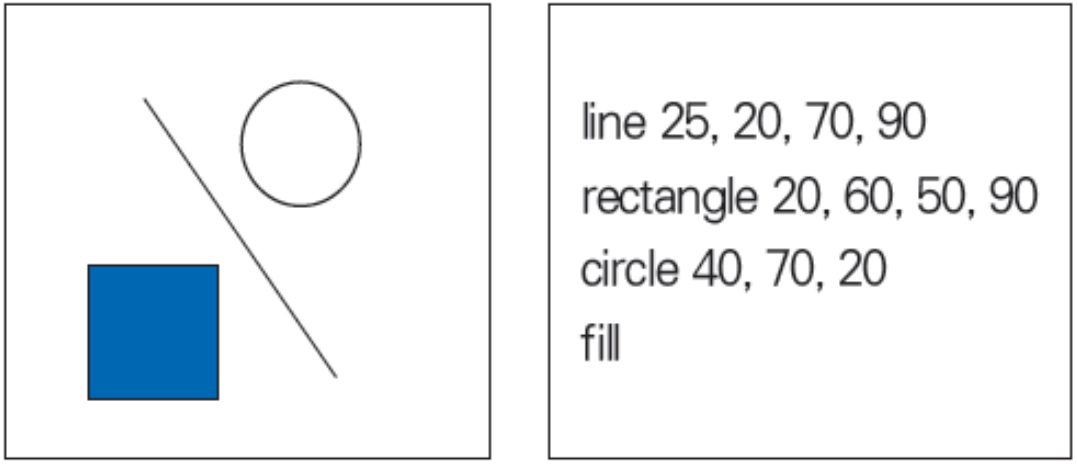
- 하나의 색상을 RGB 3가지 채널로 구분하여 표현한 예
  - 빨간색 Red
    - Red: $255_{(10)} = 11111111_{(2)} = FF_{(16)}$
    - Green: $0_{(10)} = 00000000_{(2)} = 00_{(16)}$
    - Blue: $0_{(10)} = 00000000_{(2)} = 00_{(16)}$
  - 노란색 Yellow
    - Red: $255_{(10)} = 11111111_{(2)} = FF_{(16)}$
    - Green: $255_{(10)} = 11111111_{(2)} = FF_{(16)}$
    - Blue: $0_{(10)} = 00000000_{(2)} = 00_{(16)}$
  - 흰색 White
    - Red: $255_{(10)} = 11111111_{(2)} = FF_{(16)}$
    - Green: $255_{(10)} = 11111111_{(2)} = FF_{(16)}$
    - Blue: $255_{(10)} = 11111111_{(2)} = FF_{(16)}$

2. 그림 데이터의 표현 (p.31)

- 하나의 색상을 RGB 3가지 채널로 구분하여 표현한 예
  - 검은색 Black
    - Red: $0_{(10)} = 00000000_{(2)} = 00_{(16)}$
    - Green: $0_{(10)} = 00000000_{(2)} = 00_{(16)}$
    - Blue: $0_{(10)} = 00000000_{(2)} = 00_{(16)}$
  - 주황색 Orange
    - Red: $255_{(10)} = 11111111_{(2)} = FF_{(16)}$
    - Green: $127_{(10)} = 01111111_{(2)} = 7F_{(16)}$
    - Blue: $0_{(10)} = 00000000_{(2)} = 00_{(16)}$
  - 회색 Gray
    - Red: $128_{(10)} = 10000000_{(2)} = 80_{(16)}$
    - Green: $128_{(10)} = 10000000_{(2)} = 80_{(16)}$
    - Blue: $128_{(10)} = 10000000_{(2)} = 80_{(16)}$

2. 그림 데이터의 표현 (p.32)

- 벡터(vector) 방식
  - 도형 종류, 위치, 선 길이 등 개체의 모양 정보로 그림을 표현하는 방식, 즉 개체를 그리는 명령어를 저장하는 방식



2. 그림 데이터의 표현 (p.31)

- **비트맵 방식으로 만들어진 그림의 특징**
  - 실세계의 복잡한 형상을 표현하기에 효율적이다
    - ☞ 그림이 복잡해 지더라도 용량이 증가하지 않기 때문
  - 그림 속 개체의 크기나 색상 변경이 어렵다
    - ☞ 개체를 이루는 모든 픽셀의 색상 정보를 변경해야 되기 때문
  - 확대 시 계단 현상이 발생한다

2. 그림 데이터의 표현 (p.32)

- 벡터 방식으로 만들어진 그림의 특징
  - 실세계의 복잡한 형상을 표현하기에 비효율적이다
    - ☞ 그림이 복잡해질수록 용량이 증가하기 때문
  - 그림 속 개체의 크기나 색상 변경이 쉽다
    - ☞ 개체의 모양 정보만 변경하면 되기 때문
  - 확대 시 계단 현상이 발생하지 않는다

2. 그림 데이터의 표현 (p.32)

- **비트맵 방식 그림 파일 형식**
  - 비압축
    - RAW(디지털카메라로 생성한 것)
    - BMP(그래픽 소프트웨어로 생성한 것)
  - 손실 압축
    - JPEG(Joint Photographic Experts Group)
  - 무손실 압축
    - GIF(Graphics Interchange Format)
    - PNG(Portable Network Graphics)

원본

가공한 것

2. 그림 데이터의 표현 (p.32)

- RAW
  - 1990년대 초반부터 사용
  - 디지털카메라 사진을 저장하기 위해 개발된 비압축 그림 파일
  - 제조사마다 저장되는 파일 형식 이름이 다름
    예) CRW(캐논), SRW(삼성), SRF(소니), GPR(고프로)

- BMP
  - Microsoft에서 개발 (1988)
  - 윈도우에서 사용하기 위해 개발된 비압축 그림 파일
  - 그림판, 포토샵, 김프 등 그림 편집 소프트웨어에서 생성

2. 그림 데이터의 표현 (p.32)

- JPEG(=JPG)
  - JPEG에서 개발 (1992)
  - RAW나 BMP 등 고해상도 그림 파일을 압축하기 위해 개발
  - 손실 압축을 하여 압축률이 높지만 품질이 우수함
  - 1,600만 색상 표현이 가능

[RAW]

[JPEG]

2. 그림 데이터의 표현 (p.32)

- GIF
  - CompuServe에서 개발 (1987)
  - 네트워크 상에서 빠른 전송을 위해 개발
  - 무손실 압축을 하여 압축률이 낮음
  - 표현 가능한 최대 색상 수는 256색
  - 움직이는 그림의 표현이 가능
  - 투명 배경 지원

GIF

2. 그림 데이터의 표현 (p.32)

- PNG
  - W3C(웹 표준개발 국제 컨소시움)에서 개발 (1996)
  - 무손실 압축을 하여 압축률이 낮고, 품질이 우수함
  - 1,600만 색상 표현이 가능
  - 투명 배경 지원

2. 그림 데이터의 표현 (p.32)

파일 형식	압축방식	색상수	투명 배경
BMP	압축하지 않음 (용량이 큼)	1,600만 컬러 (트루컬러)	지원 안 함 (기본적으로 지원하지 않으며 32비트 BMP에서 지원할 수 있지만 일부 프로그램에서만 인식)
JPEG	손실 압축	1,600만 컬러 (트루컬러)	지원 안 함
GIF	무손실 압축	256 컬러	지원
PNG	무손실 압축	1,600만 컬러 (트루컬러)	지원

2 디지털 정보의 연산

1. 문자 데이터의 표현
2. 그림 데이터의 표현
3. **소리 데이터의 표현**
4. 동영상 데이터의 표현

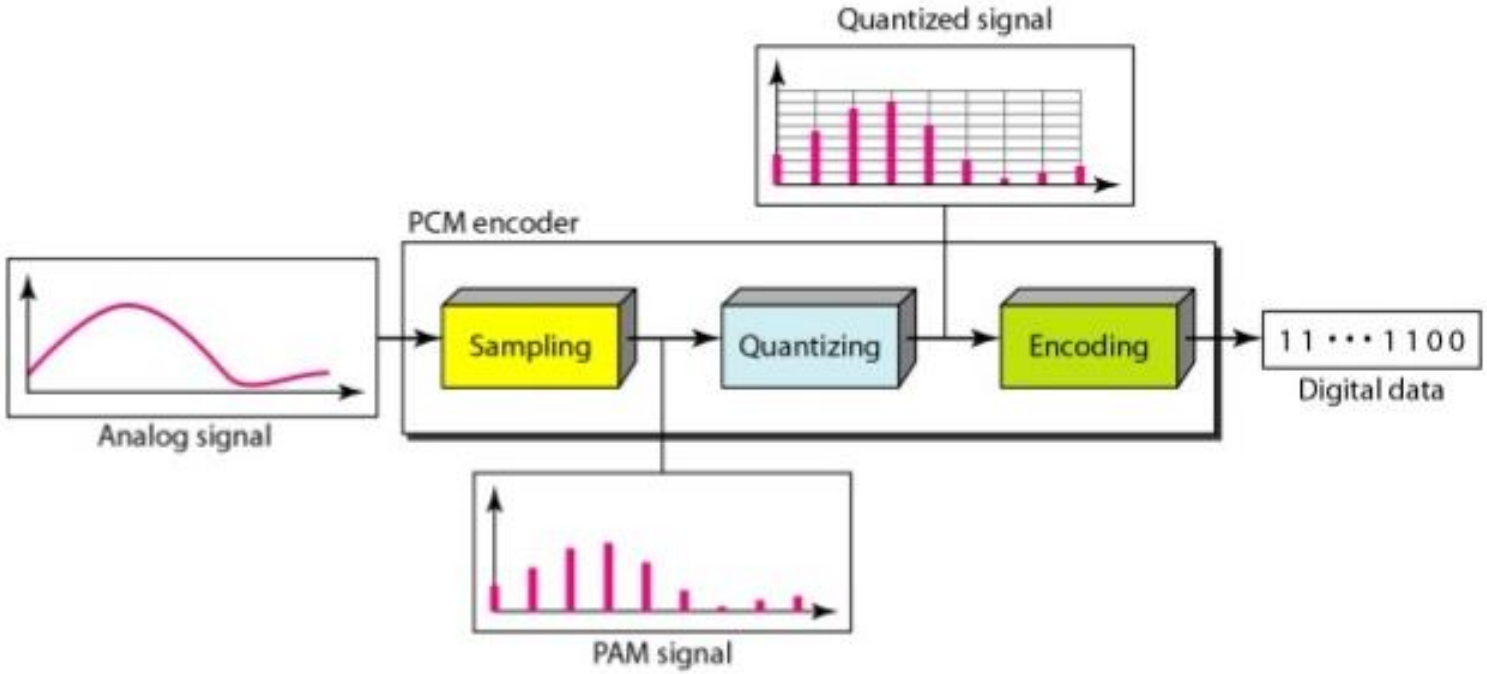
3. 소리 데이터의 표현

〈 학습 목표 〉

- 컴퓨터 시스템에서 처리하는 소리 데이터의 표현 원리를 설명할 수 있다.

3. 소리 데이터의 표현 (p.34)

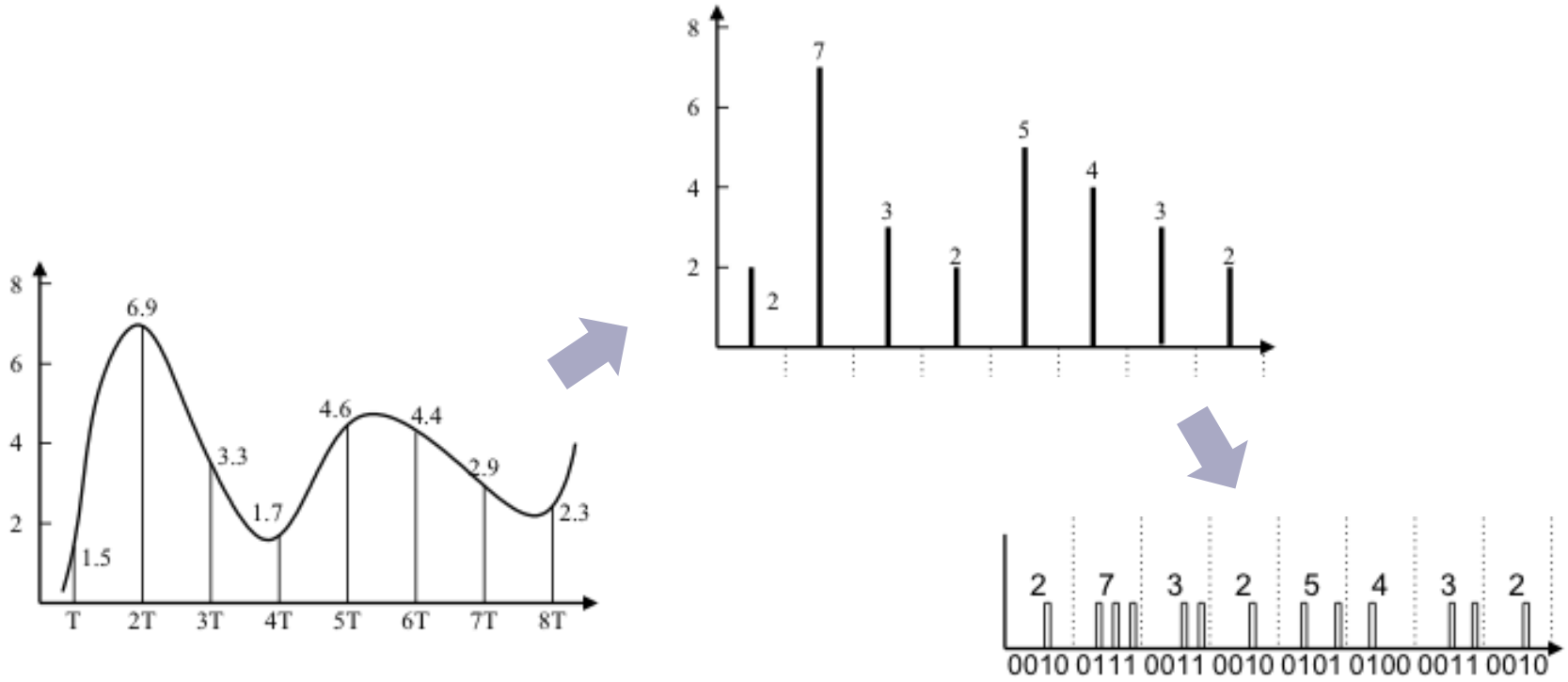
- 아날로그 소리를 디지털 파일로 만드는 과정



[PCM (Pulse Code Modulation)]

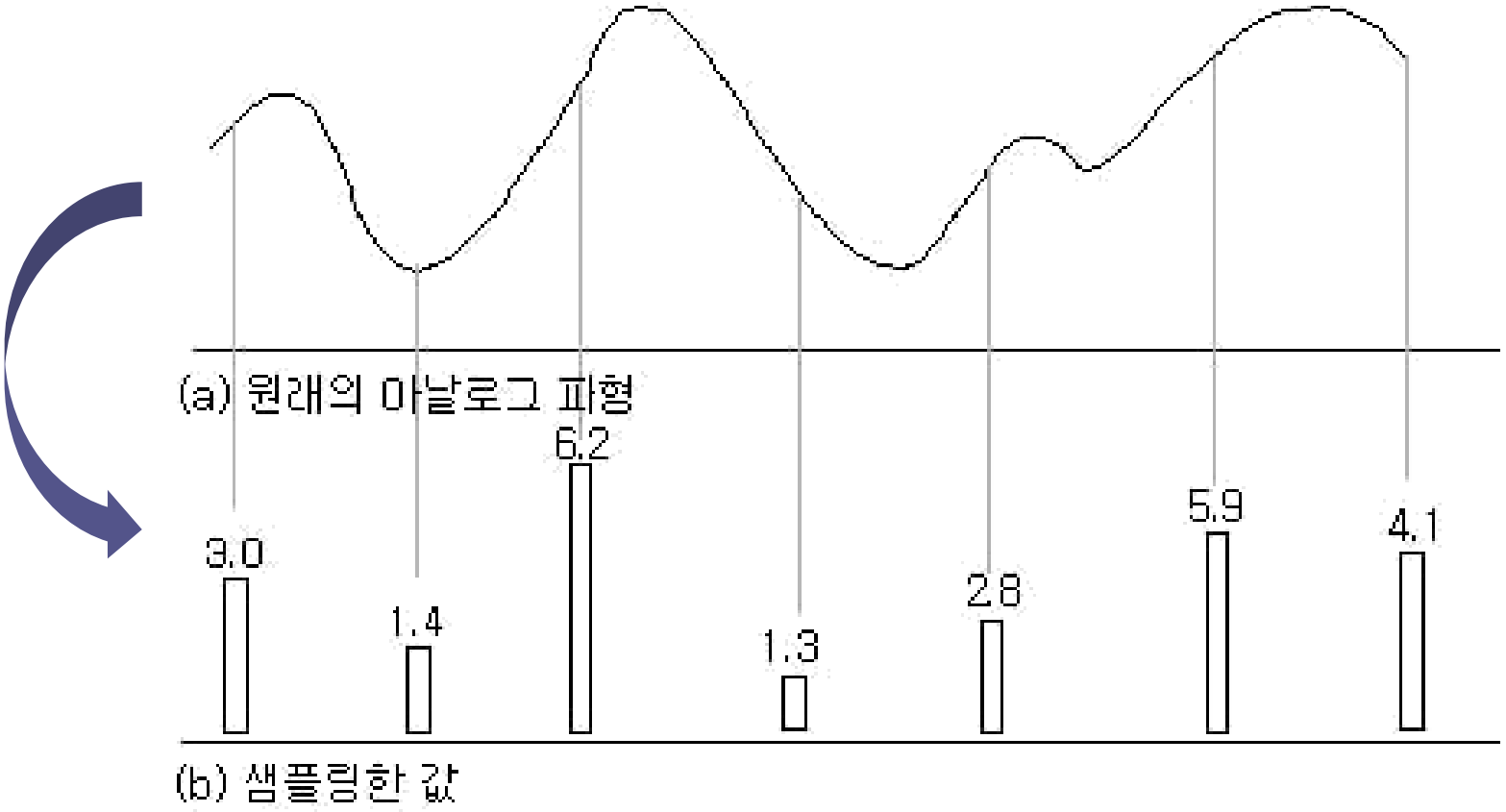
3. 소리 데이터의 표현 (p.34)

- PCM 과정
  - ① 표본화(sampling)
  - ② 양자화(quantization)
  - ③ 부호화(encoding)



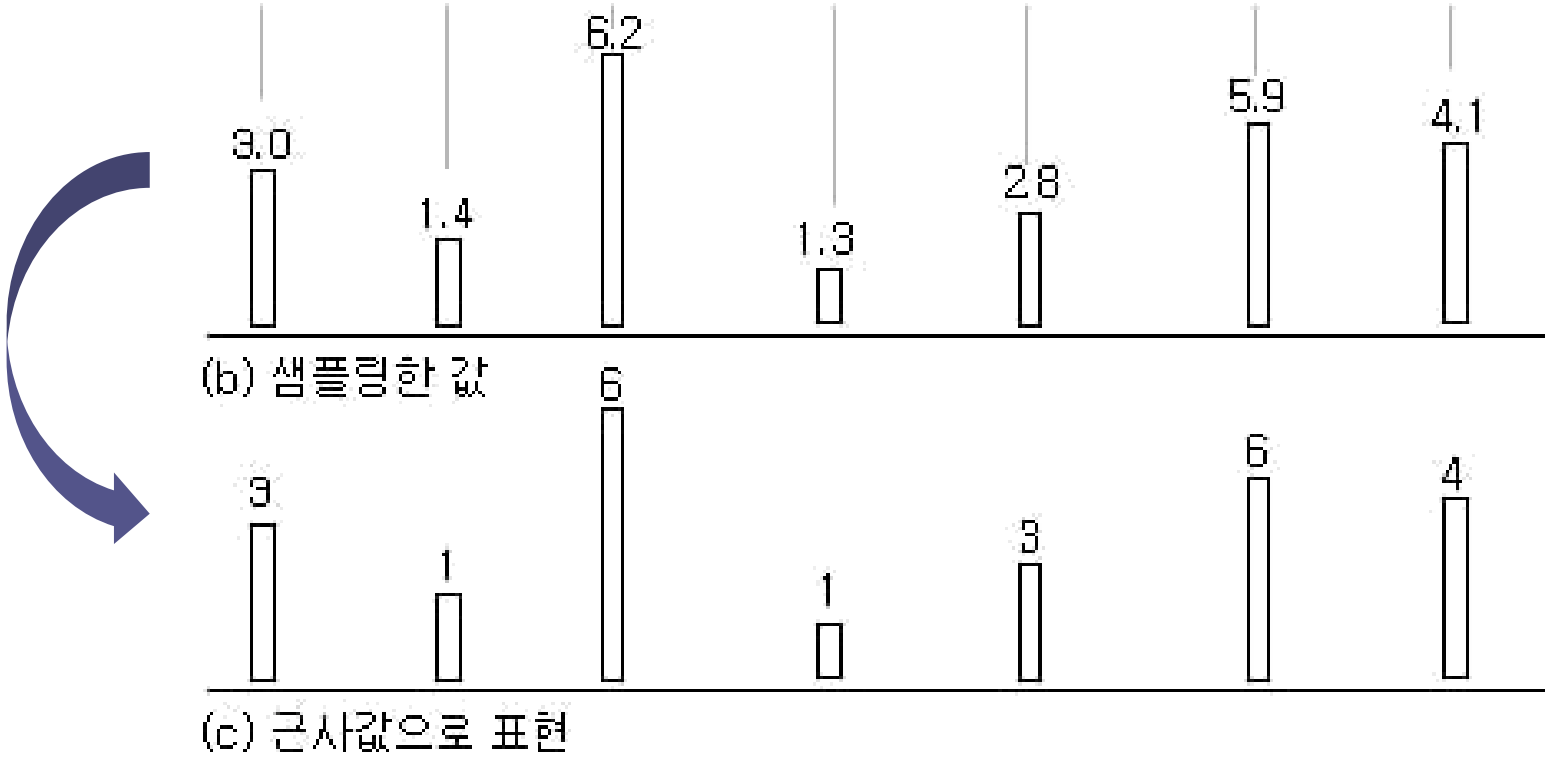
3. 소리 데이터의 표현 (p.34)

- 표본화(sampling) : (a) → (b)
  - 아날로그 신호의 진폭 값을 일정한 간격으로 측정하는 것



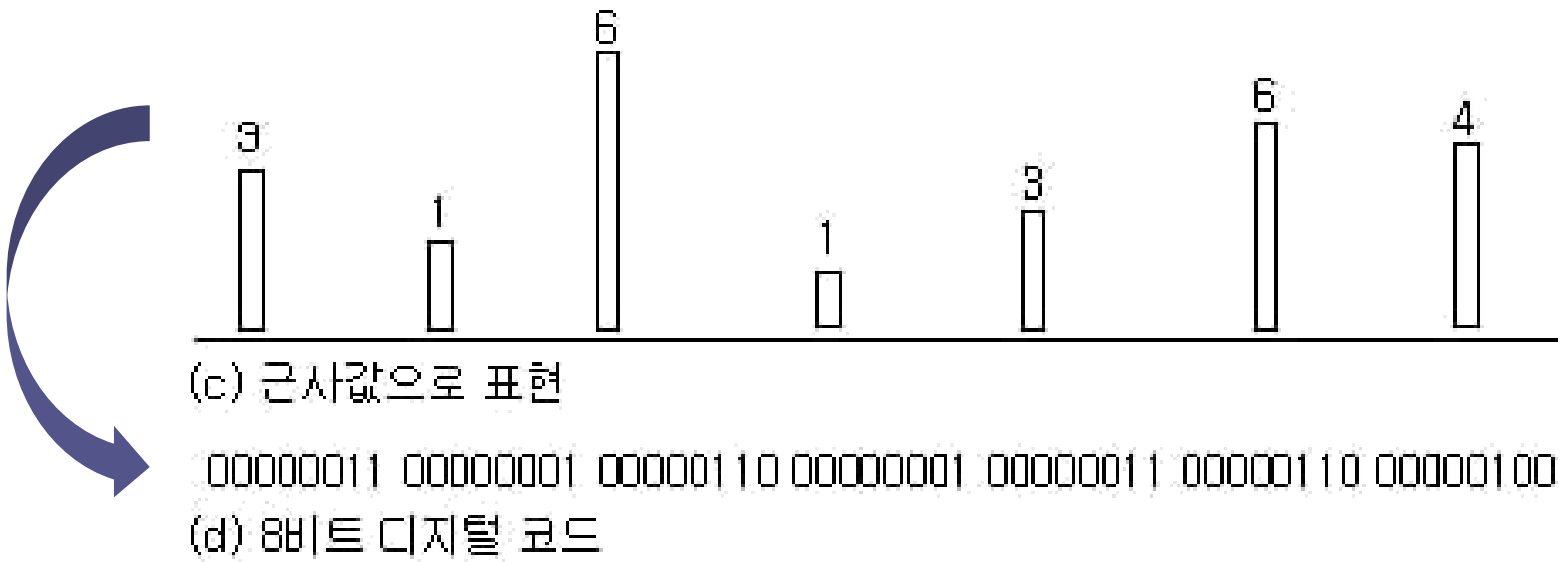
3. 소리 데이터의 표현 (p.34)

- 양자화(quantization) : (b) → (c)
  - 실숫값으로 측정된 진폭 값을 가까운 정숫값으로 표현하는 것



3. 소리 데이터의 표현 (p.34)

- 부호화(encoding) : (c) → (d)
  - 정숫값으로 표현된 값을 이진수로 변환하는 것



3. 소리 데이터의 표현 (p.34)

- **초당 샘플 수**
  - 1초 동안 측정한 아날로그 신호의 진폭 값 수
  - 표본화 비율(sampling rate)이라고도 함

- **샘플당 비트 수**
  - 진폭 값 하나를 저장할 때 사용하는 비트 수
  - 양자화 비트 수(quantization bit depth)이라고도 함

소리 파일의 음질은

초당 샘플 수와 샘플당 비트 수에 의해 결정

3. 소리 데이터의 표현 (p.34)

- **소리 파일의 크기**

초당 샘플 수 × 샘플당 비트 수 × 시간(초) × 채널 수

오디오 파일의 채널 수란?

하나의 소리 파일 안에서
독립적으로 재생되는 소리 수

Q. 계산 결과 값의 단위는?

A. 비트(bit)

채널 수가 많으면
보다 공간감(현장감) 있는
소리를 들을 수 있음

Q. 단위를 바이트(byte)로 바꾸려면?

A. 8로 나눔

[문제] 1초에 44,100개의 샘플을 만드는 비율로 16비트 크기의 샘플을 4분 동안 생성하였다. 저장된 소리 파일 원본의 크기는 몇 바이트인가? (단, 소리 파일은 스테레오(2채널)로 생성되었다.)

(풀이)

소리 파일 크기 =

초당 샘플 수 × 샘플당 비트 수 × 시간(초) × 채널 수

44,100 × 16 × 240 × 2

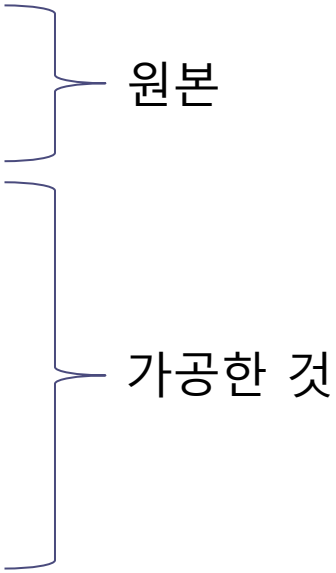
= 338,688,000 (비트)

☞ 338,688,000 ÷ 8 = 42,336,000 (바이트)

≒ 42M (바이트)

3. 소리 데이터의 표현 (p.35)

- **소리 파일 형식**
  - 비압축
    - WAV (Waveform Audio Format)
  - 손실 압축
    - MP3 (MPEG-1 Audio Layer-3)
    - AAC (Advanced Audio Coding)
  - 무손실 압축
    - FLAC (Free Lossless Audio Codec)



3. 소리 데이터의 표현 (p.35)

- WAV
  - Microsoft와 IBM이 공동 개발 (1991)
  - 윈도우에서 사용하기 위해 개발한 비압축 소리 파일
  - 오디오 CD의 표준 오디오 형식
    - ☞ 샘플당 16비트로, 44.1kHz로 샘플링된 2채널 오디오

비압축 파일인 WAV는

용량이 커서 관리나 활용이 불편하기 때문에

다양한 압축 파일 형식이 존재함

3. 소리 데이터의 표현 (p.35)

- MP3
  - MPEG(Moving Picture Experts Group)에서 개발 (1993)
  - WAV를 약 1/10~1/12로 압축한 것
  - CD 수준의 음질

- AAC
  - MPEG, AT&T, 소니, 돌비 연구소에서 공동 개발 (1997)
  - MP3를 개선하여 MP3보다 압축률이 높음

AAC는

애플 뮤직, 유튜브 뮤직, 스포티파이 등의 음원 스트리밍 서비스에서 많이 사용

* 스포티파이는 기본적으로 Ogg Vorbis를 사용하지만, iOS에서는 AAC 형식으로 스트리밍함

3. 소리 데이터의 표현 (p.35)

- FLAC
  - Xiph.org에서 개발 (2001)
  - WAV를 약 4/5~1/2로 압축한 것

많은 기기 및 소프트웨어에서 사용되는

무손실 압축계의 mp3

:: 코덱(codec) ::

- 코덱
  - Coder + Decoder
  - 음성이나 영상 자료를 압축(인코딩)하고 해제(디코딩)하는 방식 또는 이를 위한 하드웨어나 소프트웨어

- 종류
  - MP3, WMA, AAC, FLAC, ALAC, H.264, DivX, …

오디오/비디오 파일의 경우,

코덱(codec)과 포맷(format)은 동일한 의미