- 2. 데이터 표현
  - 1.1. 진법 [25]
    - 1) 수와 숫자
      - 수(number)는 그 수를 의미하는 기호인 숫자(numeric character)로 나타낸다.
    - 2) 진법
      - 용어
        - 진법
          - 수를 숫자로 나타내는 방법으로 숫자의 위치에 따라 가중치(weight)를 부여하는 방법이다.
        - 가중치(weight)
          - 가중치는 기수(radix 또는 base)의 승수(거듭제곱)를 이용한다.
        - 기수(radix, base)
          - 기수(radix, base)는 2 이상의 양의 정수다.
        - r진법
          - 기수가 r(r >= 2)인 경우의 진법
        - r진수
          - r 진법으로 표현된 수
          - r개의 숫자 (0, 1, 2, ..., r-1)로 표현
        - 기수r
          - r진수임을 나타내기 위해 r진수 오른쪽 아래 기수 r을 표기
      - r진수 N은 정수부분이 n자리, 소수점 이하 부분이 m자리라 할 때 다음과 같이 표현됨
        - a_k(a의 아래첨자 k) 는 진법에서 사용하는 숫자로, 각 자리의 계수
        - $0 \le a_k < r$ 이어야 함
      - 여러가지 진법에 따른 수 표현
        - 16진수의 경우 10개의 숫자와 나머지 6개의 숫자(A,B,C,D,E,F)는 영문자에서 빌려쓰고 있음
    - 3) 진수변환
      - (1) r진수의 10진수 변환
        - 16진수 9AB.C를 10진수로 변환하시오. 풀이 $9AB.C_{16} = 9\times16^2 + 10\times16^1 + 11\times16^0 + 12\times16^{-1}$ $= 2304 + 160 + 11 + 0.75$ $= 2475.75$
      - (2) 10진수의 r진수 변환
        - 증명[29-30]
        - 문제풀이
      - (3) 2진수와 2^n진수 상호 변환
        - 2진수를 구성하는 각 자릿수는 비트(bit : binary digit)로 표현한다.
        - n개의 비트는 2^n개의 수를 표현할 수 있다.
          - 2진수의 각 비트를 소수점 중심으로 2비트씩 묶으면 4개의 수를 표현할 수 있는 숫자 조합(00, 01, 10, 11)을 얻을 수 있다.
            - 주어진 2진수를 4개의 숫자로 수를 표현하는 4진법으로 해석할 수 있다.
        - 2진수의 2^n진수 변환
          - 예제
            - n비트씩 묶을 때 부족한 비트 수만큼 0이 있다고 가정하고 반드시 필요한 수만큼씩 묶어서 해석해야 함
        - 2^n진수(n>=2)의 2진수 변환
          - 2^n진수의 각 자리의 수에 대응하는 n비트의 2진수로 모두 바꾸면 된다.
      - 4) 기타 변환
        - r진수를 s진수로 변환하는 경우 (단, r!=s, r!=10, s!=10)
          - 1번 변환방법으로 r진수를 10진수로 변환한다음 2번 변환방법으로 다시 s진수로 변환한다.
  - 1.1. 진법 (2)
    - 4) 산술연산 (arithmetic operations)
      - 진수 r인 수에 대한 산술연산(arithmetic operations)은 10진수의 산술연산과 같다. 다만 r개의 숫자만 사용할 수 있음에 주의해야 한다.
      - 두 개의 2진수에 대한 가산, 감산, 승산, 제산
        - 두 개의 2진수에 대한 가산, 감산, 승산, 제산의 예는 다음과 같다.

- 가산
  - 두 개의 2진수 가산은 모든 자리의 합의 결과가 0이거나 1이 된다는 것을 제외하고는 10진수의 경우와 같은 방법으로 계산하며, 주저인 자리에서 발생하는 올림수(carry)는 바로 윗자리에서 합산한다.
- 감산
  - 주어진 자리 숫자에서의 내림수(borrow)가 피감수의 해당 자리에 2를 더해 주는 것을 제외하고는 10진수$_{10}$의 경우와 같다
- 승산
  - 승수의 숫자가 항상 0이거나 1이므로 부분 곱셈의 결과는 피승수와 같거나 0이 되며, 이들을 자릿수에 맞추어 모두 더해 주면 된다.
- 제산
  - 피제수에서 제수를 계속 빼서 (피제수가 제수보다 작아지거나 0이 될 때까지), 그때까지 뺀 횟수가 몫이 되고, 빼고 남은 값이 나머지가 된다.

8진법, 16진법 또는 다른 r진법의 산술연산

- 8진법$_8$, 16진법$_{16}$ 또는 다른 진법의 산술 연산을 위해서는 두 개의 한 자릿수의 합과 곱의 값을 알 수 있는 표가 필요하다. 예를 들어 4진수$_4$의 가산과 감산을 위해서는 (표 2.2)와 같은 산술 연산표를 이용하면 간편하다.

r진법의 두 수를 가산하는 보다 쉬운 방법

- 주어진 자리의 두 숫자를 10진수로 바꾸어 10진수 덧셈을 하고, 그 결과를 다시 r진법의 합과 올림수로 변환
- 예제
  - 진수가 r인 두 수의 승산은 10진법으로 산술연산하고, 그때마다의 중간 결과를 r진법으로 변환함으로써 이루어진다.
- 예제 2.13 [37]
  - 여기서 오른쪽 계산과정은 왼쪽의 두 8진수$_8$의 각 곱셈 결과를 10진수$_{10}$로 나타내고 다시 8진수$_8$로 나타낸 것이다. 가장 오른쪽의 8진수$_8$에서 ○표를 한 숫자들은 각 자리에서의 올림수(carry)를 의미한다. 예를 들면, $(4\times4)_8 = (20)_8$에서 왼쪽 숫자 2는 $(4\times7)_8$의 곱셈에 더해질 올림수이고, 마지막 자리 숫자 0만 8진수 부분 곱셈의 해당 자리에 놓이게 되는 숫자이다.

1.2. 보수 [38]

개요

- 보수의 개념, 보수를 이용한 r진수의 감산
- 2진수를 다루는 컴퓨터에서의 감산이 보수를 이용하여 가산만으로도 가능함

1) 보수의 개념

r진수$_{(r)}$ N을 r의 보수 (r's complement)와 (r−1)의 보수 ((r−1)'s complement) 등 두 가지 보수가 있다. 예를 들어, 2진수$_2$에는 2의 보수와 1의 보수가 있고, 10진수$_{10}$에는 10의 보수와 9의 보수가 있다.

(1) r의 보수

- 정수부분 n개의 자리로 구성된 r진수 N에 대한 r의 보수는 (식 2.2)와 같이 정의
  - 예를 들어, - 10진수$_{10}$ 35.34의 10의 보수는 $10^2 - 35.34 = 64.66$이고, - 2진수$_2$ 110111의 2의 보수는 $2^6 - (110111)_2 = (1000000_2 - 110111_2) = 001001_2$이다.
  - 위의 공식과 다르게 (r-1)의 보수를 구한 다음 가장 낮은 자리에 1을 더하여 구할 수도 있다
    - 즉, 10진수 678의 9의 보수는 $10^3 - 10^0 - 678 = 999 - 678 = 321$ 이므로 678의 10의 보수는 321 + 1 = 322가 된다. 2진수 1110의 1의 보수는 1110의 1을 0으로, 0은 1로 바꿈으로써 0001을 구한다. 따라서 1110_2의 2의 보수는 0001 + 1 = 0010이 된다.

(2) (r-1)의 보수

- 정수부분이 n개의 자리로 구성되고 소수점 아래가 m개의 자리로 구성된 r진수 N에 대한 (r-1)의보수는 다음과 같다.
  - N에 대한 (r−1)의 보수 = r^n − r^(−m) − N
    - 즉, $r^n - r^{-m}$에서 수 N을 빼면 (r-1)의 보수가 얻어진다. 예를 들면, 10진수 35.34에 대한 9의 보수는 $10^2 - 10^{-2} - 35.34 = 99.99 - 35.34 = 64.65$ 이고, 2진수 1101.11의 1의 보수는 $(2^4 - 2^{-2} - 1101.11)_2 = (1111.11 - 1101.11)_2 = (0010.00)_2$
      - 일반적으로 2진수의 1의 보수는 2진수의 0을 1로, 1은 0으로 서로 바꾸면 쉽게 구할 수 있다.
- r의 보수와 (r-1)의 보수의 관계
  - r의 보수 = (r-1)의 보수 + 가장 낮은 자리의 1
    - 즉, 정수에서 2의 보수는 1의 보수에 1을 더한 것과 같다.
    - 위의 공식과 다르게 (r-1)의 보수를 구한 다음 가장 낮은 자리에 1을 더하여 구할 수도 있다

2) 보수를 이용한 감산

감산을 보수와 가산으로 처리할 수 있는 이론적 근거

- 즉, 감수(subtrahend)를 부호(sign)를 보함하여 r의 보수를 한 다음 피감수(minuend)에 가산한다.
  - 10진수 감산 923-678

- 923−678=245인데 이것을 678의 10의 보수 322를 이용하여 923+322=1245를 구하고, 여기서 올림수 1을 무시하면 동일한 답 245를 구할 수 있다. 그러나 이 경우 678의 10의 보수는 1000−678=322로 구해야 하므로 여전히 감산을 이용해야 한다.
- 2진수 감산 1011_2-0101_2
  - 1011_2에 0101_2의 2의 보수를 구해 더해 준 다음 올림수만 무시하면 구할 수 있다. 그런데 여기서 0101_2의 2의 보수는 먼저 0101_2의 1의 보수를 구한 다음 그 결과에 1만 더해 주면 구할 수 있다. 그리고 0101_2의 1의 보수는 앞에서 설명한 것처럼 0은 1로, 1은 0으로 바꿈으로써 구할 수 있다.

1.3. 부호 있는 2진수[41]

개요

- 수를 표현하는 방법 중 양수뿐만 아니라 음수를 표현하는 방법
  - 수
    - 부호
    - 소수점
    - 수의 크기
- 소수점의 위치 정하는 방법에 따른 수 표현방법
  - 고정 소수점(fixed point)
    - 소수점의 위치가 고정
  - 부동 소수점(floating point)
    - 소수점의 위치가 상대적으로 이동
  - 이 항에서는 고정소수점 표현 방법에 따라 양의 정수, 영, 음의 정수를 표현하는 방법만 다룬다.
- 부호 있는 정수를 2진수로 표현하기
  - 우선 표현하고자 하는 수를 몇 비트로 표현할 것인지를 정해야 한다.
  - 4비트를 이용하여 부호 있는 2진수를 세 가지로 표현한 것
    - 어떤 표현방법을 사용하더라도 양수를 표현하는 2진수는 동일하지만, 음수를 표현하는 2진수는 각각 다르다.

1) 부호 있는 절대치 표현

- 부호 있는 절대치 표현방법
  - 수를 나타낼 때, 수의 크기(절대치)와 부호를 나타내는 비트(0 혹은 1) 구성하여 표현
  - 10진수 5를 4자리 2진수로 표현하는 경우
    - 맨 왼쪽에 4번째 비트는 양수를 나타내기 위해 0으로 설정되고, 나머지 비트는 5의 절대치를 나타내기 위해 사용
      - (0101)_2
  - 10진수 -5를 4자리 2진수로 표현하는 경우
    - 맨 왼쪽에 있는 4번째 비트는 음수를 나타내기 위해 1로 설정, 나머지 비트는 -5의 절대치를 나타내기 위해 사용한다.
      - (1101)_2
        - 2진수 4자리를 이용하여 부호 있는 절대치 표현으로 나타낼 수 있는 10진수의 범위는 -7 ~ +7이다. (1자리는 부호용)
        - 일반적으로 부호 있는 절대치 표현방법의 경우 2진수 k자리를 이용하여 나타낼 수 있는 10진수의 범위는 $-(2^{k-1}-1) \sim +(2^{k-1}-1)$이다.

2) 부호 있는 1의 보수 표현

- 양수
  - 부호 있는 절대치 표현방법을 따름
    - 10진수 5를 4자리 2진수로 표현하는 경우 5는 맨 왼쪽 비트(4번째 비트)를 양수를 나타내기 위해 0으로 설정하고, 나머지 비트는 5를 나타내기 위해 사용된다. 따라서 결과는 $(0101)_2$가 된다.
- 음수
  - 부호를 제외한 그 수에 대한 1의 보수를 이용
    - 10진수 −5를 4자리 부호 있는 1의 보수로 표현하는 경우 먼저 부호를 제외한 5에 대한 1의 보수를 구한다. 즉, 10진수 −5를 4자리 부호 있는 1의 보수로 표현하면 $(1111_2 - 0101_2) = (1010)_2$가 된다.
- 2진수 k자리를 이용하여 부호 있는 1의 보수로 나타낼 수 있는 10진수의 범위는 $-(2^{k-1}-1) \sim +(2^{k-1}-1)$이다.

3) 부호 있는 2의 보수 표현

- 양수
  - 부호 있는 절대치 표현방법을 따름
    - 10진수 5를 4자리 2진수로 표현하는 경우, 5는 맨 왼쪽(4번째) 비트를 양수를 나타내기 위해 0으로 설정되고, 나머지 비트는 5를 나타내기 위해 사용된다. 따라서 결과값은 $(0101)_2$가 된다.
- 음수

- 부호를 제외한 그 수에 대한 2의 보수를 이용하여 표현
  - 10진수 −5를 4자리 부호 있는 2의 보수로 표현하는 경우, 부호를 제외한 5에 대한 2의 보수를 구하면 $10000_2 - 0101_2$가 된다. 따라서 결과값은 $(1011)_2$가 된다.
- 2진수 4자리를 이용하여 부호 있는 2의 보수 표현으로 나타낼 수 있는 10진수의 범위는 <표 2.3>과 같이 −8 ~ +7 이다.
- 2의 보수 절대치 표현방법의 경우, 2진수 k자리를 이용하여 나타낼 수 있는 10진수의 범위는 $-2^{k-1} \sim +(2^{k-1} - 1)$ 이다.
  - 앞의 세 가지 표현방법 중 부호 있는 2의 보수 표현방법에서는 0의 표현이 항상 한 가지인 데 비해 나머지 두 방법은 +0과 −0의 두 가지이다. 따라서 부호 있는 2의 보수 표현방법은 항상 나머지 두 방법에 비해 동일한 비트 수가 주어지는 경우 하나의 수를 더 표현할 수 있다.

- 예제
  - 예제 2.14
    - 부호 있는 절대치 표현
      - 부호 비트가 1이므로 음의 정수이고, 크기 부분은 다음과 같이 1178로 계산되므로 실제 저장된 정수는 −1178이다.
    - 부호 있는 1의 보수 표현 방법
      - 먼저 비트열 **1000010010011010_2**의 1의 보수를 구한다. 이를 위해 각 비트에 대해 보수를 (0은 1로, 1은 0으로) 취하면 **0111010110011010_2**이 되고, 위 방식처럼 자릿수를 이용하여 계산하면 **+31589**이 된다. 따라서 실제 저장된 정수는 **−31589**이다.
    - 부호 있는 2의 보수 표현
      - 해당 비트열에 대해 2의 보수를 구한다. 이것은 우선 1의 보수를 구한 다음 **1**만큼 더하면 된다. 위의 계산을 이용하면 2의 보수는 **0111010110011011_2**이 되고, 그 값은 **+315900**이므로, 실제 저장된 정수는 **−315900**이다.
        - 따라서 저장된 정수가 음수인 경우는 부호 있는 2의 보수 표현이 부호 있는 1의 보수 표현보다 **1**만큼 더 작은 값을 나타낸다.
  - 예제 2-15.

2.2. 디지털 코드 [45]

- 개요
  - 디지털 코드(digital code)
    - 사람이 사용하는 데이터를 표현하기 위해 비트를 필요한 수만큼 묶어서 각 묶음에 유일한 의미를 부여하는 것
    - 10진 코드
    - 영숫자 코드
- 1. 10진 코드
  - 10진 코드(decimal code)
    - 사람의 입장에서 진수변환이라는 부담을 줄이기 위해 컴퓨터 내부에 10개의 숫자를 2진코드로 만들어 활용
  - 임의의 r진수를 2진 코드로 나타내려면 $2^{k-1} < r \le 2^k$ 을 만족하는 k개의 비트가 필요해진다.
    - r = 10인 경우를 고려하면, k = 4가 되므로 10진수를 2진 코드로 나타내려면 4개의 비트가 필요하다.
  - 10진수를 표현하는 대표적인 2진 코드 유형
    - BCD 8421 코드
    - BCD 2421 코드
    - 84-2-1 코드
    - 3초과(excess -3) 코드
    - 그레이 코드
  - 1) BCD 8421 코드
    - BCD 8421 코드는 단순히 BCD 코드라고도 부르는데, 2진화 10진(binary-coded decimal: BCD) 코드의 형식으로 10진수 한 자리를 2진수 4비트로 표시한 것이며, 비트의 위치에 따라 8, 4, 2, 1(2^3, 2^2, 2^1, 2^0)의 가중치(weight)가 할당된다.
      - 가중치 코드(weighted code)
        - 각 비트 위치에 가중치가 할당되어 코드화된 조합에서 모두 1의 가중치 합을 더함으로써 10진수 한 자리 수치를 구할 수 있도록 설계된 코드
    - 주의 : 여러 자리로 구성되는 10진수 전체가 2진법으로 표현되는 것은 아니다.
      - 10진수 14는 BCD 코드로 표현하면 (0001 0100)_BCD이고 2진법으로 표현하면 1110_2이라는 점에 유의해야 한다.
      - 14 = (0001 0100)_BCD = $1110_2 \neq$ (1110)_BCD
      - BCD 코드에서는 1110이라는 비트 조합을 코드로 사용하지 않음
    - 대부분의 컴퓨터에서는 10진수를 BCD 코드로 변환하여 저장한다.
      - BCD 코드로 표현되는 2진수에 대한 산술연산이 필요하다.

- 산술연산 중에서 덧셈에 대한 연산이 가장 중요하다.
  - 산술연산 중 뺄셈, 곱셉, 나눗셈은 덧셈과 보수를 이용하여 모두 계산되기 때문이다.
- BCD 코드에 대한 덧셈과정
  - 1. 2진수의 덧셈 규칙에 따라 두 수를 더한다.
  - 2. 연산 결과 4비트의 2진수 값이 9 이하이면 그 값을 그대로 사용한다.
  - 3. 연산 결과 4비트의 2진수의 값이 9보다 크면 BCD 코드에서는 사용되지 않는 값이므로, 이 경우 그 결과값에 6(0110)을 더한다.
    - 6만큼 더하는 이유? 사용되지 않는 비트의 조합이 6개 있기 때문
    - 만약 올림수가 발생하면 한 자리 높은 자릿수에 올림수를 더한다.
  - 예제

2) BCD 2421 코드

- BCD 2421 코드는 BCD 8421 코드와 마찬가지로 10진수를 각 자릿수마다 4비트로 구성하여 표현하되 가중치를 2, 4, 2, 1로 하는 가중치 코드이다.
- 예시
  - 10진법의 5는 가중치를 2, 4, 2, 1로 하여 $5 = 1\times2 + 0\times4 + 1\times2 + 1\times1$ 이므로 2421 코드로 5는 $(1011)_2$로 표현된다.
  - $5 = 0\times2 + 1\times4 + 0\times2 + 1\times1$ 이므로 5를 $(0101)_2$로 표현할 수도 있겠으나, 동일한 수 하나에 대해 오직 하나의 코드만 대응시켜야 하므로 비트 조합 $(0101)_2$는 BCD 2421 코드에서 사용되지 않는다.
- 장점 : 자보수 코드(self - complement code)
  - 10진수에 대한 9의 보수를 코드에서 0은 1로, 1은 0으로 바꾸어서 쉽게 구할 수 있다.
    - 예를 들어, 10진수 28은 BCD 2421 코드로 $(0010\ 1110)_{2421}$이고, 28의 9의 보수는 $99 - 28 = 71$이므로 BCD 2421 코드로는 $(1101\ 0001)_{2421}$이 된다. 여기서 $(0010\ 1110)_{2421}$에서 0을 1로, 1을 0으로 바꾸면 9의 보수인 71의 BCD 2421 코드를 쉽게 구할 수 있다.

3) BCD 84-2-1 코드

- BCD 8421 코드와 마찬가지로 각 자릿수마다 4비트로 구성하여 표현하되 가중치를 8, 4, -2, -1로 한다.
  - 예) 10진수 2는 84-2-1 코드로 0110으로 표현된다.
- 자보수 코드
  - 10진수에 대한 9의 보수를 코드에서 0을 1로, 1을 0으로 바꿔서 구할 수 있음

4) 3초과 코드

- 3초과(excess-3) 코드는 비가중치 코드(unweighted code)이며, BCD 8421 코드에 10진수 3에 해당하는 2진수 0011을 각각 더해서 만든다.
  - 예를 들어, 10진수 13은 각 자리에 3을 더해서 46을 만든 후 BCD 8421 코드에서 찾으면 0100 0110이 된다.
- 자보수 코드

5) 그레이 코드

- 그레이 코드(gray code)는 가중치를 갖고 있지 않은 비가중치 코드이며, 산술연산에는 부적합하지만 한 숫자에서 다음 숫자로 이동될 때 오직 한 비트만 변하는 특징이 있으므로 입출력장치, A/D 변환기 등에 많이 쓰이고 있다.
- 그레이 코드는 4비트 2진수로부터 다음과 같이 정의된다.
  - ① 맨 왼쪽의 자리에 있는 2진 숫자를 그대로 가져온다.
  - ② 맨 왼쪽에서부터 시작하여 오른쪽의 인접 자릿수에 있는 2진 숫자와 합해진 올림수는 무시하여 코드의 해당 자리 숫자를 만들어 낸다(이것은 XOR 연산으로도 가능).
  - ③ 오른쪽 인접 자릿수가 없을 때까지 ②번을 반복한다.
- 예시

예제

2.2. 디지털 코드 (2) [50]

2. 영숫자 코드

영숫자 코드(alphanumeric code)

- 10진 숫자 10개와 영문자 26개, 특수문자 및 기호 등으로 구성되어 최소한 36개 이상의 원소로 구성됨
- 분류
  - 영문자의 대문자와 소문자를 구별하지 않는 경우
    - 영문자의 대문자와 소문자를 구별하지 않고 오직 영문자 26개와 10진 숫자 10개만으로 구성되는 영숫자 코드 집합인 경우, 정확히 36개의 영숫자만 표현하면 되므로 비트 6개로써 이들을 모두 나타낼 수 있다. ($2^{k-1} < 36 \le 2^k$을 만족하는 k를 구해 보라).
  - 영문자의 대문자와 소문자를 구별하는 경우

- 특수문자 및 기호 등을 포함시키고 영문자의 대문자와 소문자를 구별해야 한다면 최소한 7개 이상의 비트가 필요하다. 따라서 일반적으로 영숫자 코드 집합을 만들기 위해서 각 코드는 7비트 혹은 8비트로 구성되어야 한다.

- ASCII, EBCDIC 코드

1) ASCII(American Standard Code for Information Interchange) 코드
- 하나의 영숫자 코드가 7비트로 구성되어 있으며, 전체적인 코드의 수는 128(=2^7)개
  - 10진 숫자: '0' ~ '9' (10개)
    - 10진 숫자에 대한 코드는 전체 비트 중 맨 왼쪽 3비트 (비트 위치 6, 5, 4)가 011이고 나머지 비트(비트 위치 3, 2, 1, 0)는 BCD 코드와 동일한 값이다.
      - 입력된 10진 숫자의 ASCII 코드는 맨 왼쪽의 3비트(011)만 제거하면 BCD 코드와 동일해지므로 BCD 코드에 대한 연산을 직접적으로 할 수 있다.
  - 영문자 : 대문자 'A'~'Z'(26개), 소문자 'a'~'z'(26개)
  - 기호문자 : 'SP'(space), '!', '"', '#', '$', '%' 등(33개)
  - 제어문자 : NUL, SOH, STX, ETX, EOT, DEL 등(33개)
    - 제어문자(control character)는 데이터 전송 시 필요한 제어기능과 컴퓨터 또는 외부장치의 기능을 제어하고자 할 때 사용한다.
    - DEL(111 1111)을 제외한 모든 제어문자는 맨 왼쪽 3비트가 000과 001로 되어 있다
- 특징
  - 전체 7비트 + 1비트로 구성된 패리티 비트(parity bit)를 포함시켜 8비트로 전송한다.
    - 대부분의 컴퓨터가 정보의 기본단위로 바이트(byte = 8 bit)를 사용하기 때문에, 패리티 비트를 포함한 ASCII 코드를 사용하는 것이 매우 효율적이다.

2) EBCDIC 코드
- EBCDIC(Extended Binary Coded Decimal Interchange Code)
  - 하나의 영숫자 코드가 8비트로 구성되어 있으며, 총 256개의 문자 코드를 구성할 수 있으나, 실제로 사용되는 문자 코드의 수는 ASCII 코드와 동일한 128개이다.
  - 10진 숫자 코드는 전체 비트 중 맨 왼쪽 4비트가 1111이고 나머지 비트는 BCD 코드와 동일한 값을 가지고 있다.
    - 맨 왼쪽의 4비트(1111)만 제거하면 BCD 코드와 동일해지므로 BCD 코드에 대한 연산을 직접적으로 할 수 있다.
- 단점
  - 전체 구성 비트 수가 8비트이기 때문에 에러 검출을 위한 패리티 비트를 추가하면 총 9비트가 된다. 그런데 대부분의 컴퓨터가 정보의 기본 단위로 바이트를 사용하기 때문에 이 경우 ASCII 코드에 비해 데이터 전송 시 비효율적일 수 있다.

3) 유니코드
- 유니코드(unicode)
  - 사용 중인 플랫폼, 프로그램, 언어에 관계없이 문자마다 고유한 숫자를 부여하는 방법을 제공하는 문자 인코딩 표준
    - ASCII는 알파벳만을 부호화하는 데 사용하지만 유니코드는 전 세계의 문자를 표현하는 총체적 문자 코드 체계
- 인코딩 방식
  - UTF('UCS Transformation Format')
    - 뒤에 붙는 숫자는 인코딩에 사용되는 단위의 비트 수
  - UFT-8 (8 bit), UTF-16 (16 bit), UTF-32 (32bit)
    - 공통적으로 문자를 표현하는데 4비트를 사용함
  - UTF-8
    - ASCII 모든 문자 호환, 기존 소프트웨어에서도 소프트웨어를 수정하지 않고 사용 가능
    - 유니코드 문자를 1바이트에서 4바이트 형태로 가변적으로 인코딩
  - UTF-16
    - 대행 문자(surrogate) 2개의 쌍으로 1개의 문자를 인코딩
  - UTF-32
    - 고정길이 인코딩
    - 메모리 공간에 제약이 없을 때 사용

4) 기타 영숫자 코드
- 5비트 보도 코드(Baudot code)
  - 문자 모드
  - 도형 모드
- 6비트 코드
- 천공카드 코드

- 천공카드를 이용하여 영숫자 정보를 컴퓨터에 입력시킬 때 사용되는 12비트로 구성되는 홀러리스 코드(Hollerith code)가 대표적
- 천공카드는 12행과 80열로 구성되는데, 각 열의 적절한 행이 천공된 부분은 1로 감지되며, 천공되지 않은 부분은 0으로 감지된다.
- 용어
  - 존 천공(zone punch)
  - 숫자 천공(numberic punch)