2.2 Мерење на количеството информација

- **Бит** = единица мерка за мерење на количеството информација (основна мемориска единица)
  - **binary digit = бинарна цифра**, 0 или 1
- **Бит** = количество информација содржано во одговорот на кое било прашање за кое се можни **два** еднакво веројатни одговори

2.2 Мерење на количеството информација

Пример:

- Замислен е број од 1 до 16. Колку пати треба да се постави прашањето: **„Дали бројот е поголем од X?“** за да се погоди замисленият број?
- Претпоставка: замислен е бројот 7
  - 1. Дали бројот е поголем од 8? НЕ **1,2,3,4,5,6,7,8**
  - 2. Дали бројот е поголем од 4? ДА **5,6,7,8**
  - 3. Дали бројот е поголем од 6? ДА **7,8**
  - 4. Дали бројот е поголем од 7? НЕ **7**

ДА–>1, НЕ–>0, **НЕ-ДА-ДА-НЕ –> 0110** (четири бита)

2.2 Мерење на количеството информација

- Ако се можни **n** еднакво веројатни одговори на едно прашање, количеството информација може да се запише со **k**-битови, при што важи релацијата: $\mathbf{n=2^k}$, односно $\mathbf{k=\log_2 n}$
- Во претходниот пример: $\mathbf{16=2^4}$, односно $\mathbf{4=\log_2 16}$

2.2.1 Претставување на податоците

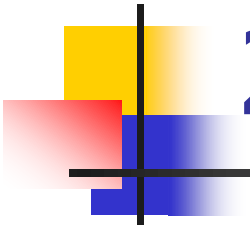
- Податоците во компјутерите се претставуваат со цифрите 0 и 1 – **бинарен** броен систем (со основа 2)
- **Декаден** броен систем (со основа 10) – десет цифри: 0,1,2,3,4,5,6,7,8,9
- **Октален** броен систем (со основа 8) – осум цифри: 0,1,2,3,4,5,6,7
- **Хексадецимален** броен систем (со основа 16) – шеснаесет цифри: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

2.2.2 Зошто бинарен систем?

- Дигитална информација може да се запомни правејќи разлика помеѓу вредностите на некоја непрекината физичка величина, како на пример електричен напон, јачина на струја, итн.
- Колку повеќе вредности треба да се разликуваат, толку е помало раздвојувањето помеѓу соседните вредности, а помала е и доверливоста на меморијата
- Кај **бинарниот броен систем**, потребно е да се разликуваат само две вредности – најдоверлива метода за кодирање на дигитални информации

2.2.2 Зошто бинарен систем?

- Секоја информација може да се претстави со цифрите 0 и 1, односно со низа од битови
- Операциите со овие две цифри се многу едноставни
- Цифрите 0 и 1 во електрониката репрезентираат две вредности на напон
  - Пример: 1 –> 2.4-5 V, 0 –> 0-0.8 V
- Постојат физички медиуми со особина да паметат само две состојби
  - Пример – магнетни материјали:
    - 1 –> магнетизирана точка
    - 0 –> немагнетизирана точка



2.3 Примарна меморија

- **Меморијата (memory, store, storage)** – оној дел од компјутерскиот систем во кој се чуваат (складираат) програмите и податоците

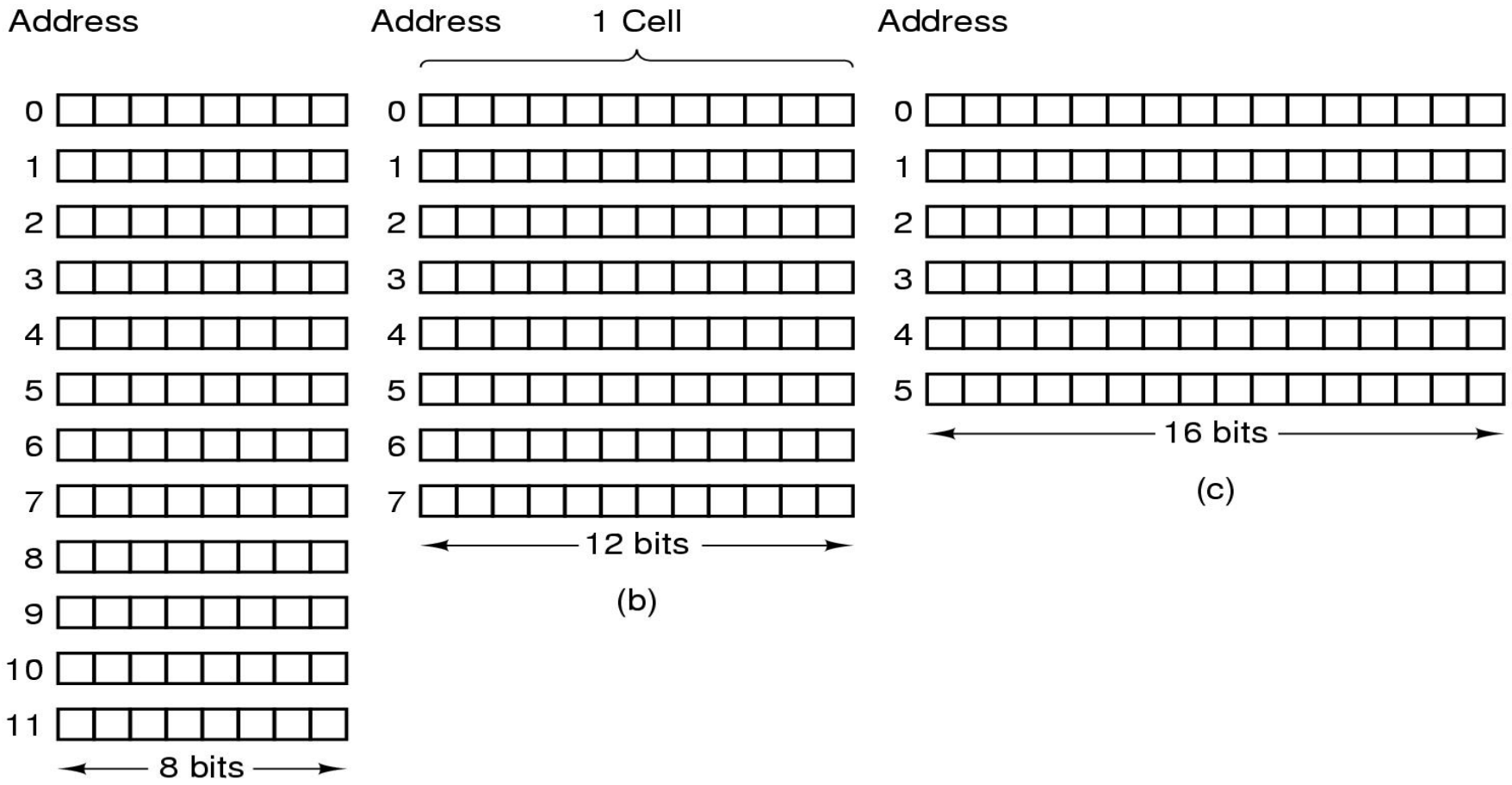
2.3.1 Мемориски адреси

- Мемориите се состојат од одреден број **ќелии (cells)** или **локации (locations)**, секоја со своја **адреса** преку која програмите можат да и се обраќаат
- **n** ќелии $\rightarrow$ адреси од **0** до **n-1**
- **k** битови $\rightarrow$ $\mathbf{2^k}$ различни комбинации
- Ако адресата е претставена со **m** битови, максималниот број на ќелии кои можат да бидат адресирани е $\mathbf{2^m}$

2.3.1 Меморискиадреси

- 8-битна ќелија претставува **бајт (byte)**
- Бајтовите се групираат во 32-битни или 64-битни **зборови (words)**
- Повеќето инструкции манипулираат со цели зборови
  - 32-битна машина има 32-битни регистри и инструкции за манипулирање (преместување, собирање, одземање, ...) со 32-битни зборови
  - 64-битна машина има 64-битни регистри и инструкции за манипулирање со 64-битни зборови

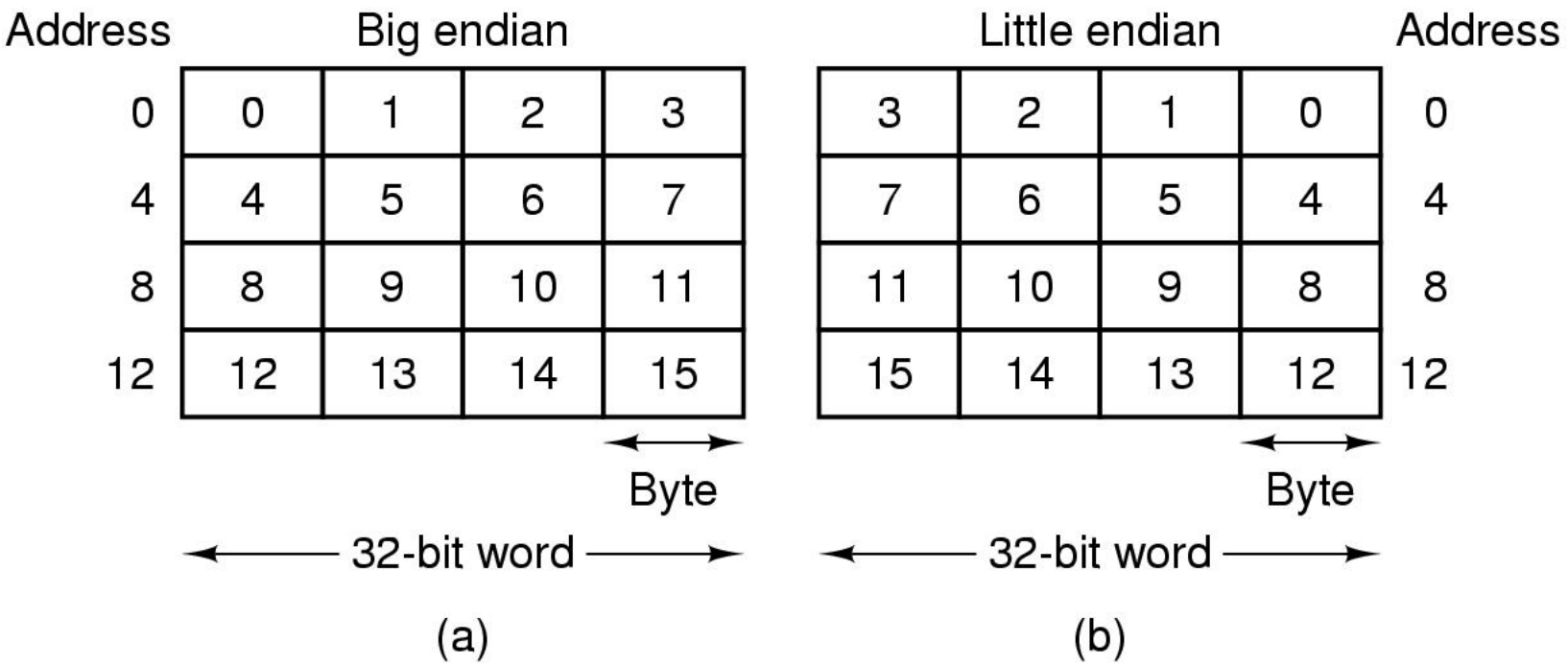
2.3.1 Мемориски адреси



2.3.2 Подредување на бајтовите

- Бајтовите во еден збор можат да бидат нумерирани на еден од следните начини:
  - одлево-надесно – **big endian** компјутер (SPARC, IBM mainframe, …)
  - оддесно-налево – **little endian** компјутер (Intel processor family, …)

2.3.2 Подредување на бајтовите



2.3.3 Кодови за корекција на грешки

- Заради заштита од повремено појавување на грешки (како резултат на промени во напонот, или од други причини), кај мемориите се применуваат:
  - кодови за **откривање** на грешки (error-detecting codes)
  - кодови за **корекција** на грешки (error-correcting codes)
- Реализација: на секој меморискиот збор, на посебен начин, му се додаваат дополнителни битови кои се проверуваат при читањето, со цел да се открие дали се појавила грешка

2.3.3 Кодови за корекција на грешки

- Нека **n**-битен **коден збор (codeword)** се состои од **m** податочни битови и **r** битови за проверка (n=m+r)
- **Hamming-ово растојание (d)** – број на бит-позиции во кои два кодни збора се разликуваат
  - Пример: 10001001, 10110001 $\rightarrow$ d=3
- Ако се состави листа од сите дозволени кодни зборови и ако во таа листа се најдат двата кодни збора чие Hamming-ово растојание е најмало, тогаш така определеното растојание е Hamming-ово растојание на целиот код

2.3.3 Кодови за корекција на грешки

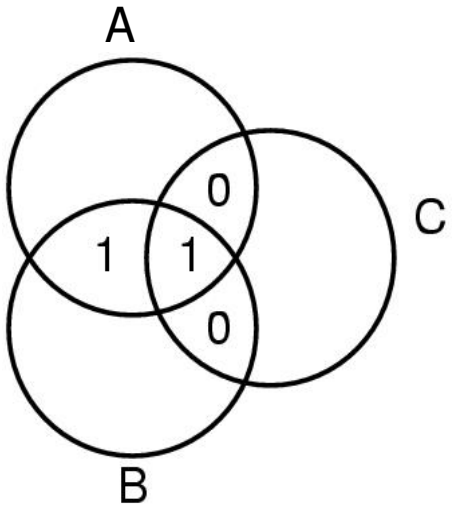
- За **откривање** на **n** едно-битни грешки, потребен е код со растојание d=n+1
  - n едно-битни грешки не можат да претворат еден валиден коден збор во друг валиден коден збор
- Пример
  - **бит за парност (parity bit)** – на податокот му се додава уште еден бит, така што бројот на единици во кодниот збор да биде парен
    - **10001001 → 100010011, 10001011 → 100010110**
  - **d=2** – може да се користи за откривање на грешка само кај еден бит (појавата на веќе 2 едно-битни грешки резултира со друг валиден коден збор)

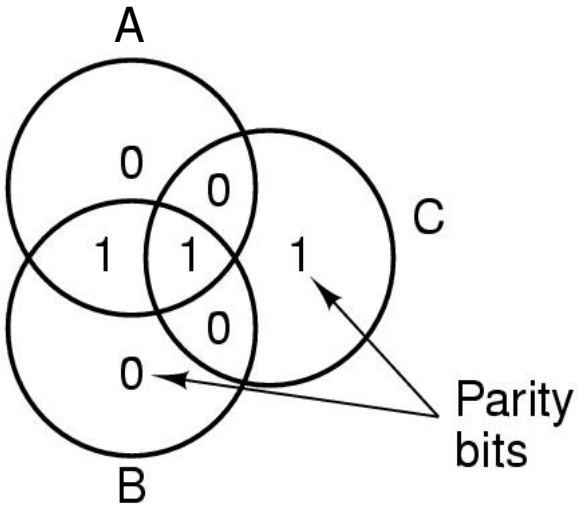
2.3.3 Кодови за корекција на грешки

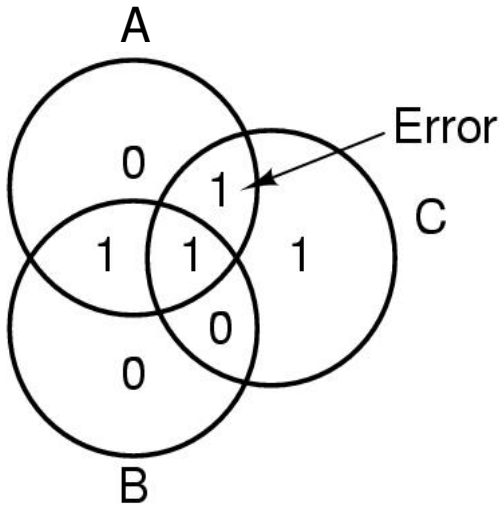
- За **корекција** на **n** едно-битни грешки, потребен е код со растојание d=2n+1
  - дозволените кодни зборови се толку далеку еден од друг, што дури и со n промени, оригиналниот коден збор е поблиску од кој и да е друг валиден збор, па може да биде еднозначно определен
- Пример
  - Нека еден код има само 4 валидни кодни зборови:
    - **0000000000, 0000011111, 1111100000, 1111111111**
  - **d=5** – може да се користи за корекција на n=2 едно-битни грешки
    - **0000000111** сигурно потекнува од **0000011111** (ако не се појавиле повеќе од две грешки!)

2.3.3 Кодови за корекција на грешки

- Графичка илустрација на идејата за код за корекција на грешки кај 4-битни зборови
  - **m**=4, **r**=3, **n**=m+r=7







2.3.3 Кодови за корекција на грешки

- Хамингов алгоритам
  - Сите битови чиј реден број е степен на бројот 2, се т.н. **битови за парност**
  - Битот со реден број **b** го проверуваат битовите со редни броеви $\mathbf{b_1, b_2, \ldots, b_j}$ за кои важи $\mathbf{b_1 + b_2 \ldots + b_j = b}$
    - Пример: битот 5 го проверуваат битовите 1 и 4 (1+4=5)
  - Ако се појавила грешка, редниот број на изменетиот бит е еднаков на **збирот од редните броеви** на битовите за парност кои даваат некоректен резултат

2.3.3 Кодови за корекција на грешки

- Пример
  - 16-битен збор + 5 битови за парност = 21-битен коден збор
    - Бит 1 ги проверува битовите: 1,3,5,7,9,11,13,15,17,19,21
    - Бит 2 ги проверува битовите: 2,3,6,7,10,11,14,15,18,19
    - Бит 4 ги проверува битовите: 4,5,6,7,12,13,14,15,20,21
    - Бит 8 ги проверува битовите: 8,9,10,11,12,13,14,15
    - Бит 16 ги проверува битовите: 16,17,18,19,20,21
  - Ако се јавила грешка кај битот со реден број 5, тогаш битовите со редни броеви 1 и 4 ќе даваат неточна информација за парноста