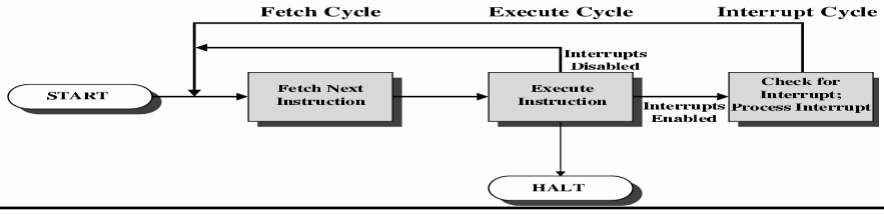
Data Representation in Computer Systems

Lecture 7

Interrupt Cycle

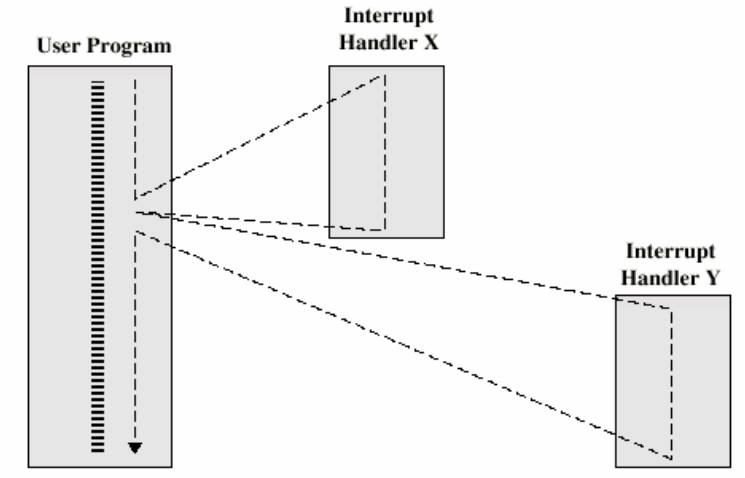
- Added to instruction cycle
- Processor checks for interrupt
  - Indicated by an interrupt signal
- If no interrupt, fetch next instruction
- If interrupt pending:
  - Suspend execution of current program
  - Save context
  - Set PC to start address of interrupt handler routine
  - Process interrupt
  - Restore context and continue interrupted program



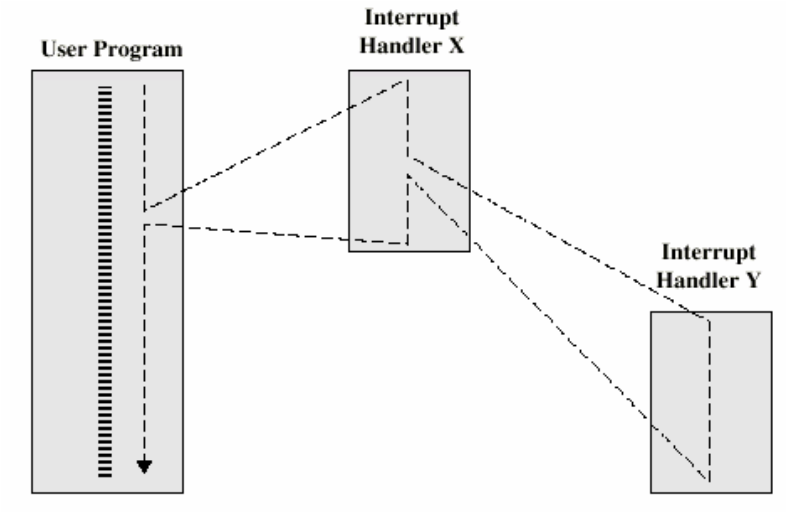
Multiple Interrupts

- Disable interrupts
  - Processor will ignore further interrupts whilst processing one interrupt
  - Interrupts remain pending and are checked after first interrupt has been processed
  - Interrupts handled in sequence as they occur
- Define priorities
  - Low priority interrupts can be interrupted by higher priority interrupts
  - When higher priority interrupt has been processed, processor returns to previous interrupt

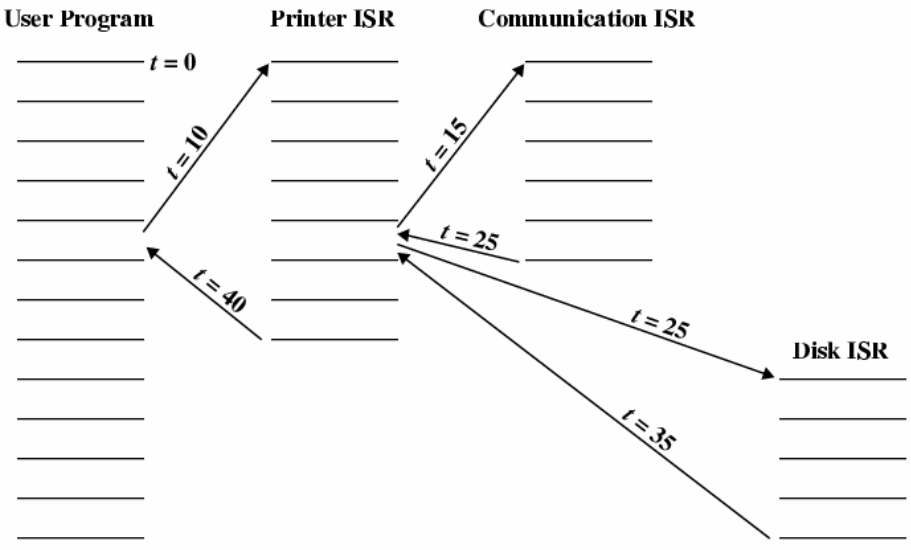
Multiple Interrupts - Sequential



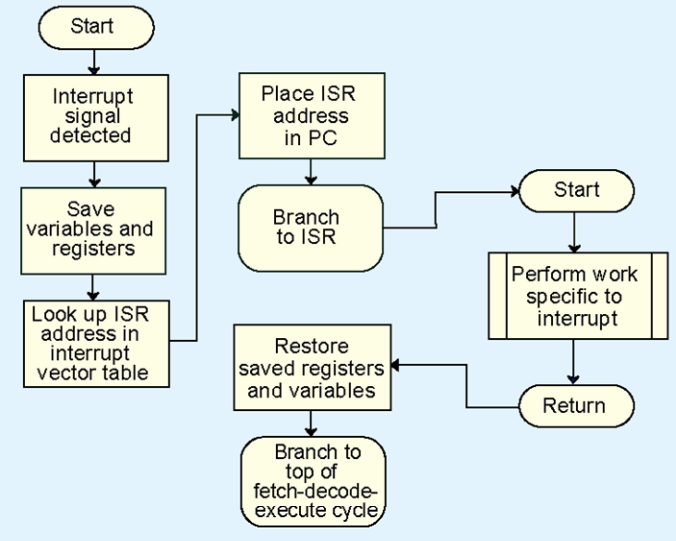
Multiple Interrupts – Nested



Time Sequence of Multiple Interrupts



Instruction Processing with interrupt



Character Codes

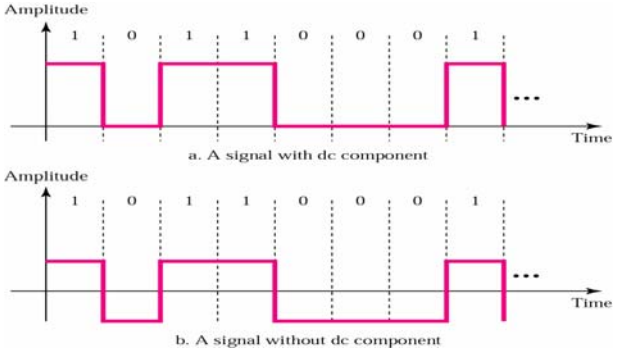
- As computers have evolved, character codes have evolved.
  - The earliest computer coding systems:
    - Binary-coded decimal (BCD) was one of these early codes.
      - It was used by IBM mainframes in the 1950s and 1960s
      - Larger computer memories and storage devices permit richer character codes.
    - Extended Binary Coded Decimal Interchange Code (EBCDIC)
      - IBM mainframe and midrange computer systems.
    - ASCII
      - American Standard Code for Information Interchange
    - Unicode
      - A 16-bit code that is downward compatible with ASCII

Encoding Techniques

- Both analog and digital information can be encoded as either analog or digital signals
- Need to be able to determine beginning and end of each bit (clocking or synchronization)
  - Digital data, digital signal
    - Simplest form, binary ones and zeros
  - Digital data, analog signal
    - Modem converts digital data to analog signal to transmit over analog line
  - Analog data, digital signal
    - Digitize analog data for digital transmission
  - Analog data, analog signal
    - No conversion (voice, telephone)

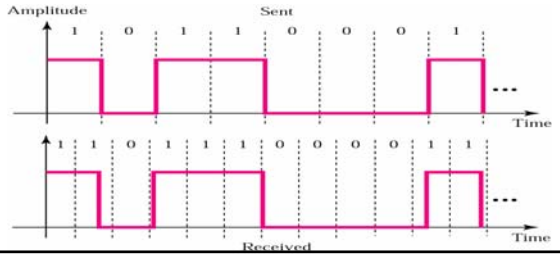
DC Components

- Some line coding scheme leave a undesirable direct-current (dc) component (zero frequency)
  - It is undesirable because
    - Transformer does not allow to pass a signal with the dc component presence
    - The distorted signal creates errors in the output
    - Dc component is extra energy present on the line and is useless



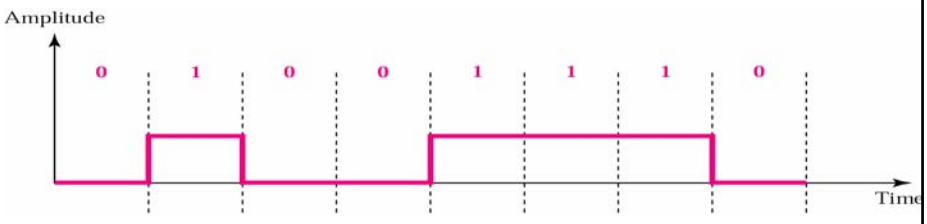
Self synchronization

- To interpret correctly what sender has sent the signal, the receiver's bit interval must corresponds to the sender's bit interval
  - If the receiver's clock is faster or slower, the bit intervals are not matched and the receiver may interpret differently
- A self synchronizing digital signal includes timing information in the signal
  - The synchronization can be achieved if there are transitions in the signal that alert the receiver to the start, middle or end of the pulse
  - If the receiver's clock is out of synch, the alerting points can reset the clock



Encoding Scheme

- Unipolar: simplest, inexpensive to implement but obsolete in use
  - It provides the concept of encoding system
  - Most encoding scheme uses to send one voltage level for zero, and another for one
  - The polarity of the pulse decides whether it is posative or negative
- Unipolar scheme uses only one polarity, that is assigned to one of the two binary states, normally the 1
  - The other state is zero voltage



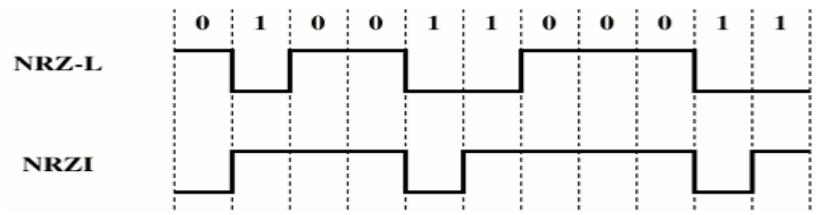
•presence of dc component (average amplitude is non zero
•Lack of synchronization (if data contains long 1's or 0's)

Encoding Schemes

- Polar encoding:
  - Uses two voltage levels, one positive and one negative
    - Due to two voltage levels average voltage level on the line is reduced and the dc component problem is eliminated
  - Nonreturn to zero (NRZ) the value of the signal is always either positive or negative
    - NRZ- L
      - ***The level of the signal is dependent upon the state of the bit***
    - NRZ – I
      - ***The signal is inverted if a 1 is encountered.***

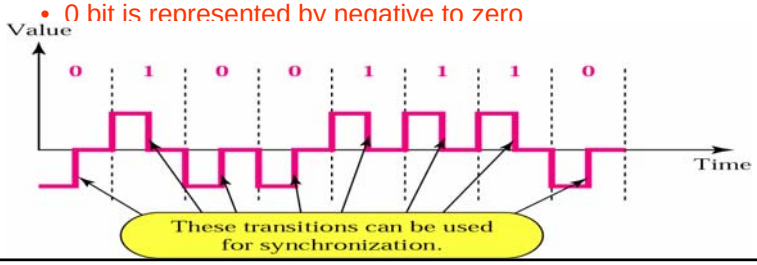
Polar: Nonreturn to Zero-Level & NRZ Invert (NRZ-L & NRZ-I)

- Two different voltages (+ and – OR + and none) for 0 and 1 bits
  - Voltage constant during bit interval
    - no transition I.e. no return to zero voltage
    - e.g. Absence of voltage for zero, constant positive voltage for one
  - More often, negative voltage for one value and positive for the other



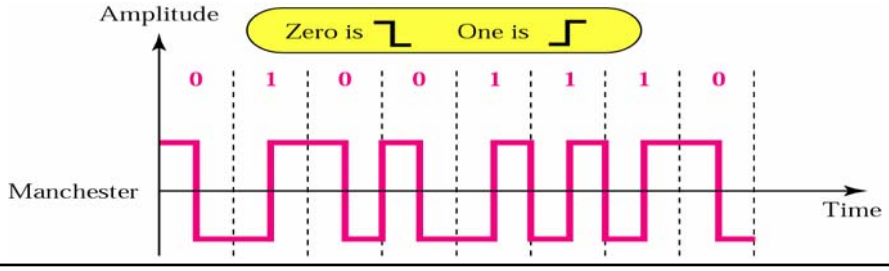
Polar: Return to Zero- RZ

- With Return to Zero (RZ) encoding, the three values to the signal can be assigned
- With RZ signal changes not between bits but during each bit
  - Like NRZ-L, a positive voltage means a 1 and a negative voltage means 0
  - With RZ , half way through each bit interval, the signal return to zero
    - 1 bit is represented by positive to zero
    - 0 bit is represented by negative to zero



Polar: Manchester encoding

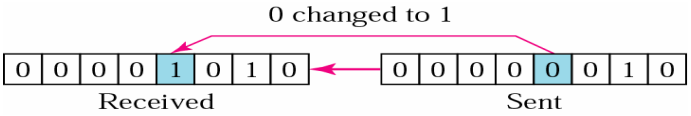
- It uses an inversion at the middle of each bit interval for synchronization and bit representation
  - A negative to positive transition represents binary 1
  - A positive to negative transition represents binary 0
- By using the signal transition for dual purpose, this encoding scheme has the same level of synchronization as of RZ, but with two levels of amplitude



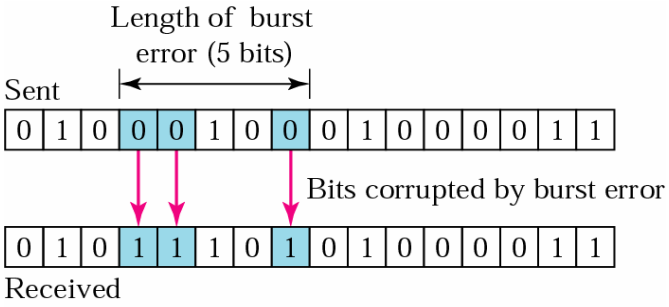
Error Detection and Correction

- It is physically impossible for any data recording or transmission medium to be 100% perfect 100% of the time over its entire expected useful life.
  - ***Data can be corrupted during transmission. For reliable communication, errors must be detected and corrected.***
- As more bits are packed onto a square centimeter of disk storage, as communications transmission speeds increase, the likelihood of error increases.
- Thus, error detection and correction is critical to accurate data transmission, storage and retrieval.
  - **Single-Bit Error**
  - **Burst Error**

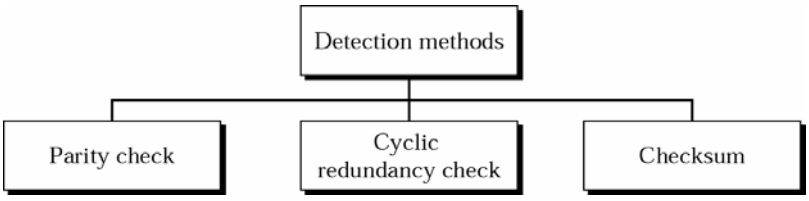
In a single-bit error, only one bit in the data unit has changed.



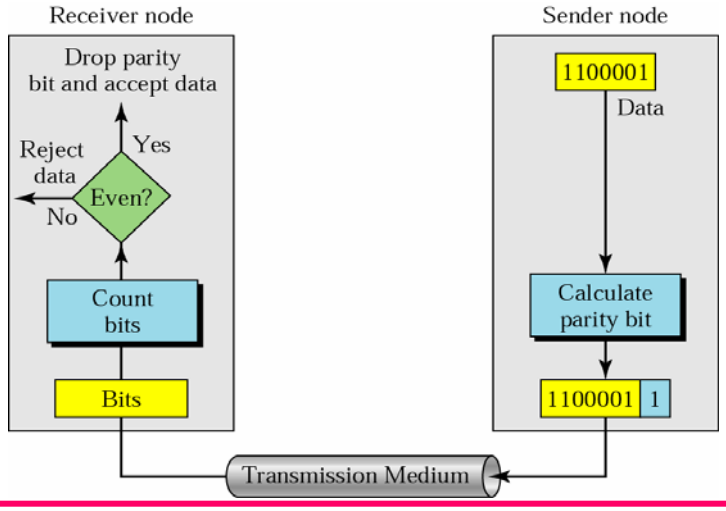
A burst error means that 2 or more bits in the data unit have changed.



Detection methods



Even-parity concept



In parity check, a parity bit is added to every data unit so that the total number of 1s is even (or odd for odd-parity).

Example 1

• Suppose the sender wants to send the word *world*. In ASCII the five characters are coded as

1110111 1101111 1110010 1101100 1100100

The following shows the actual bits sent

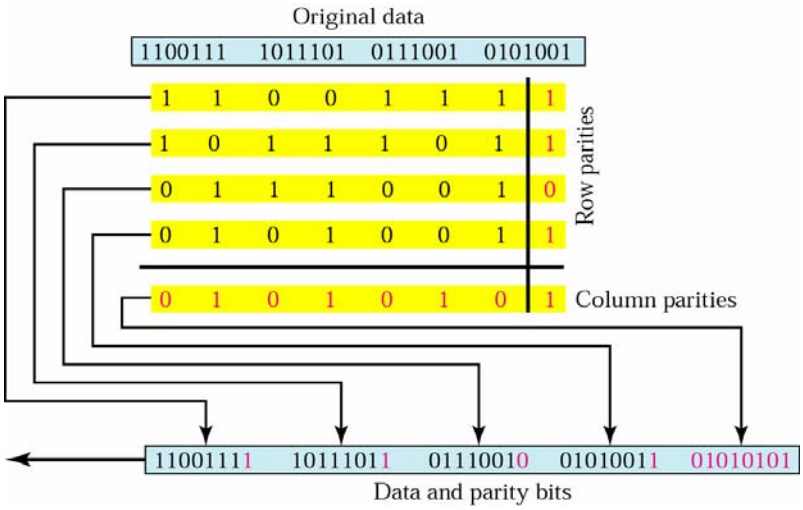
0 1110111 **0**1101111 1110010**0** **0** 1101100 **1**1100100

Now suppose the word world is received by the receiver without being corrupted in transmission.

01110111 01101111 01110010 0 1101100 11100100

The receiver counts the 1s in each character and comes up with even numbers (6, 6, 4, 4, 4). The data are accepted. It is dropped otherwise

Two-dimensional parity



Example

Suppose the following block is sent:

10101001 00111001 11011101 11100111 10101010

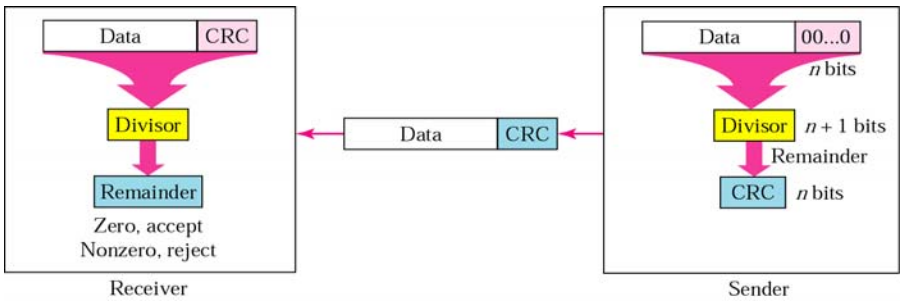
The block is hit by a burst noise of length 8, and some bits are corrupted.

10100011 10001001 11011101 11100111 10101010

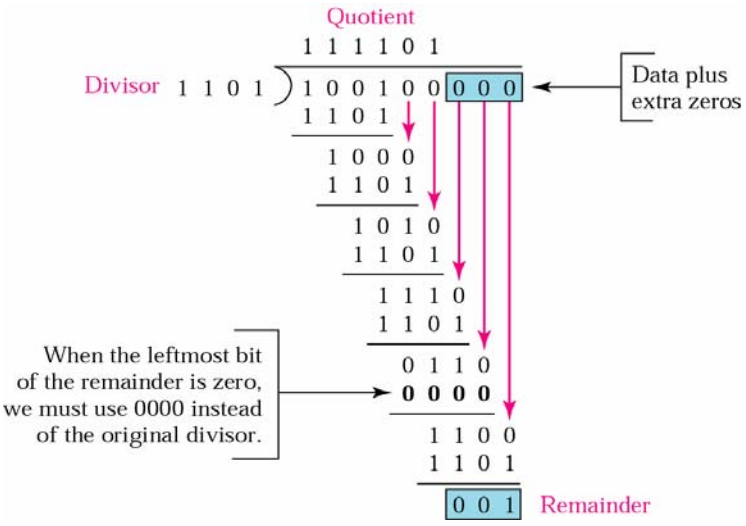
When the receiver checks the parity bits, some of the bits do not follow the even-parity rule and the whole block is discarded.

10100011 10001001 11011101 11100111 **1**0**101**0**1**0

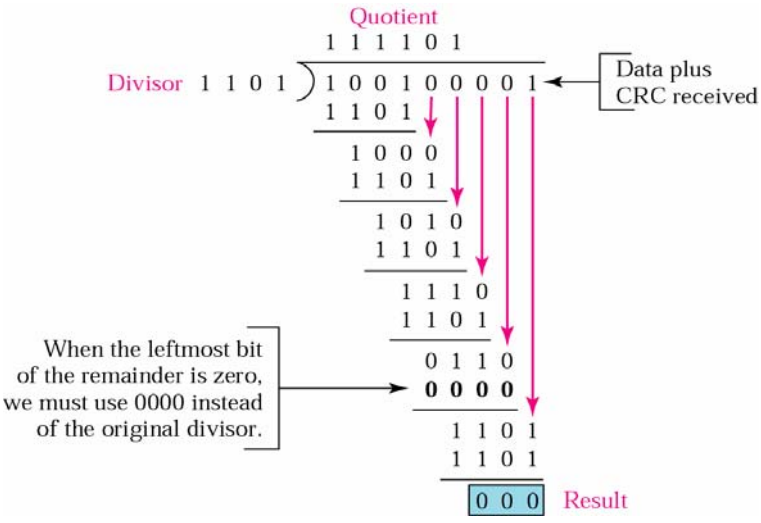
CRC generator and checker



Binary division in a CRC generator



Binary division in CRC checker



Error Detection and Correction

- Data transmission errors are easy to fix once an error is detected.
  - Just ask the sender to transmit the data again.
- In computer memory and data storage, however, this cannot be done.
  - Too often the only copy of something important is in memory or on disk.
- Thus, to provide data integrity over the long term, error *correcting* codes are required.

Error Detection and Correction

- Hamming codes:
  - Used in situations where random errors are likely to occur
    - use parity bits, also called *check bits* or *redundant bits*
    - The memory word itself consists of *m* bits, but *r* redundant bits are added to allow for error detection and/or correction
    - The final word, called a *code word*, is an *n*-bit unit containing *m* data bits and *r* check bits.
      - Code word = N bits = m bits + r bits
  - The number of bit positions in which two code words differ is called the *Hamming distance* of those two code words.

Data and redundancy bits

Number of data bits m	Number of redundancy bits r	Total bits m + r
1	2	3
2	3	5
3	3	6
4	3	7
5	4	9
6	4	10
7	4	11

Error Detection and Correction

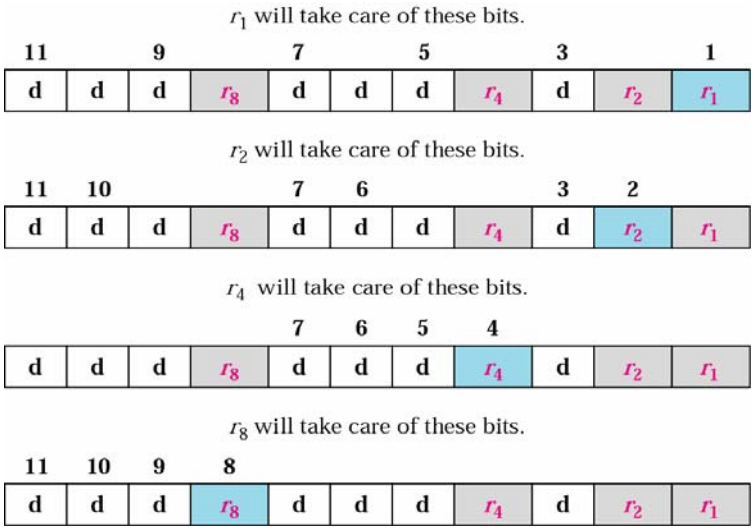
- With code words of length 12, we observe that each of the digits, 1 though 12, can be expressed in powers of 2. Thus:

$1 = 2^0$ $\quad 5 = 2^2 + 2^0$ $\quad 9 = 2^3 + 2^0$
$2 = 2^1$ $\quad 6 = 2^2 + 2^1$ $\quad 10 = 2^3 + 2^1$
$3 = 2^1 + 2^0$ $\quad 7 = 2^2 + 2^1 + 2^0$ $\quad 11 = 2^3 + 2^1 + 2^0$
$4 = 2^2$ $\quad 8 = 2^3$ $\quad 12 = 2^3 + 2^2$

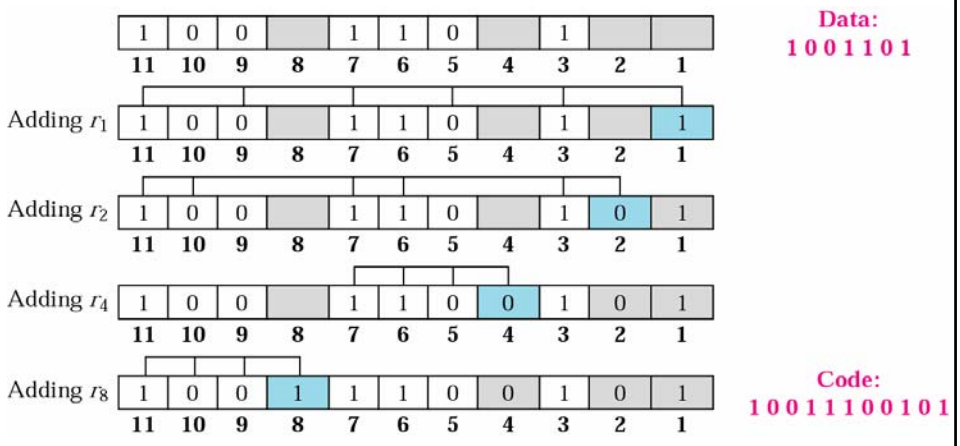
  - 1 (= 2^0) contributes to all of the odd-numbered digits.
  - 2 (= 2^1) contributes to the digits, 2, 3, 6, 7, 10, and 11
  - Bit 4 checks digits, 5, 6, 7, and 12
  - Bit 8 checks digits, 9, 10, 11, and 12.
  - . . . And so forth . . .
- We can use this idea in the creation of our check bits.

11	10	9	8	7	6	5	4	3	2	1
d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

Redundancy bits calculation



Example of redundancy bit calculation



Error detection using Hamming code