Lecture 7

Data Integrity
Internal Memory

Parity
Hamming Code
Cache
Memory

Checking Data Integrity

- In general case, you can not confirm data integrity
- Errors may be observed and sometime automatically corrected
  - Due to a fault data becomes erroneous (<u>error</u>)
    - Bit is still always 1 or 0
  - E.g., bit ”flips” in memory or in transmission
    - Memory circuit may have a <u>fault</u> (static fault)
    - Subatomic particle may flip bit in memory or in transmission (transient fault)
  - Uncorrected error may cause a <u>failure</u>
- Database inconsistency is separate topic!

$$\begin{tikzcd} \text{fault} \arrow[d] \\ \text{error} \arrow[d] \\ \text{failure} \end{tikzcd}$$

More info? → Data-base courses

Fault, Error, Failure

- *The Big Bang Theory* tv-series
  - Penny slips in bath tub?
  - Sheldon explains to doctor that the problem (fault) was no anti-skid surface in bath tub, which caused slipping (error), which in turn caused a fracture (failure). Penny said that she slipped in a tub.
- Space Shuttle

  - Faults were usually problems in programmer training
  - Error was a problem in code
    - Errors were classified (1-5) based on the failure they would cause (if executed)
  - Failure could be minor (1), or ”loss of vehicle” (5)

Protecting Data Integrity

- Faults and errors happen, even though they are rare
- Use extra bits, which allows errors to be detected and possibly to be corrected (need to know which bit was bad!)
- System does the checking automatically
  - At hardware level, e.g.,
    - Memory circuit checks, whether data has changes during storate, and maybe corrects erroneous data
    - CPU MMU checks, whether data has become erroneous during bus transit, and maybe corrects erroneous data
  - At software level, e.g.,
    - There are no errors in 4KB data packet received
- Data integrity vs. accepted risk
  - Accept bad data not found once a year? Once in 10 years?
  - Risk management

Bit level checks

National ID Nr, Hetu

- Memory circuits, buses, disks, data transmissions

Hetu: 120464-121C

(chars, not bits) ?

- How many flipped (erroneous) bits are detected?

Hetu: 1 (char)

- How many flipped bits can be automatically corrected?

Hetu: 0 (char)

- How many extra bits are needed to detect and correct erroneous bits?
  - More memory, more disk space?
  - Extra wires in bus?

Hetu: +10%

- Is error detection and correcting done at hardware or at software level?

Hetu: software

Parity bit

Detect: All one bit errors

- One extra bit for each data item
  - Byte, word, data packet (?)
- Even parity: number of 1-bits is always even
- Odd parity: number of 1-bits is always odd
- Detect: 1 bit errors
- Correct: 0 bit errors
- OK for situations, where 2 or more errors are unlikely
- Example (even parity)

0010 001 0   1000 1101 1111 001 1

Extra space needed: 12.5%   6.25%

Finnish IBAN Bank Number

Detect:

- All 1 character errors
- Most 2 character errors
- All switches of 2 chars
- Most other errors

(old)
Bank account number

500015-123

50001500000123 FI00

50001500000123 15 18 00

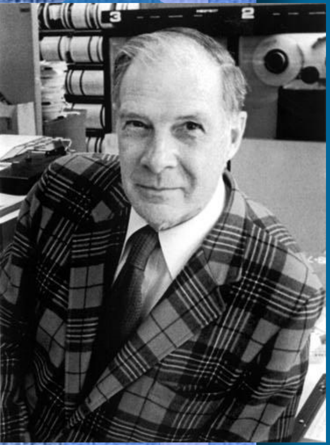
50001500000123 15 18 00 mod 97 = 61

98-61 = 37

FI$\underline{37}$ 5000 1500 0001 23

(New, IBAN)
Bank account number

Hamming Distance (R. Hamming 1950)



- In given coding system (e.g., ISO Latin-1), how many bits in any code (e.g., ´A’ = 0x41 = 0100 0001) must be flipped, that it will become to some other (any other) <u>legal</u> code?

$$\begin{tikzcd}
\text{´A’} = \text{0x41} = 0100\ 0001 \arrow[d, bend left, "\text{2 bits}"] \\
\text{´B’} = \text{0x42} = 0100\ 00\underline{10} \arrow[d, bend left, "\text{1 bit}"] \\
\text{´C’} = \text{0x43} = 0100\ 001\underline{1}
\end{tikzcd}$$

- ISO Latin-1 Hamming distance: 1
- With parity bit, Hamming distance: 2
  - What is the likelyhood for 2 bit error (vs. 1 bit error)?
  - Is it sufficiently small?

Prob{ ”2 bit error” } = (Prob{ ”1 bit error” })^2

if errors are independent

Error Correcting Hamming Code (5) (R. Hamming 1950)

Enlightening Example on the Principle

(d) Correct the error

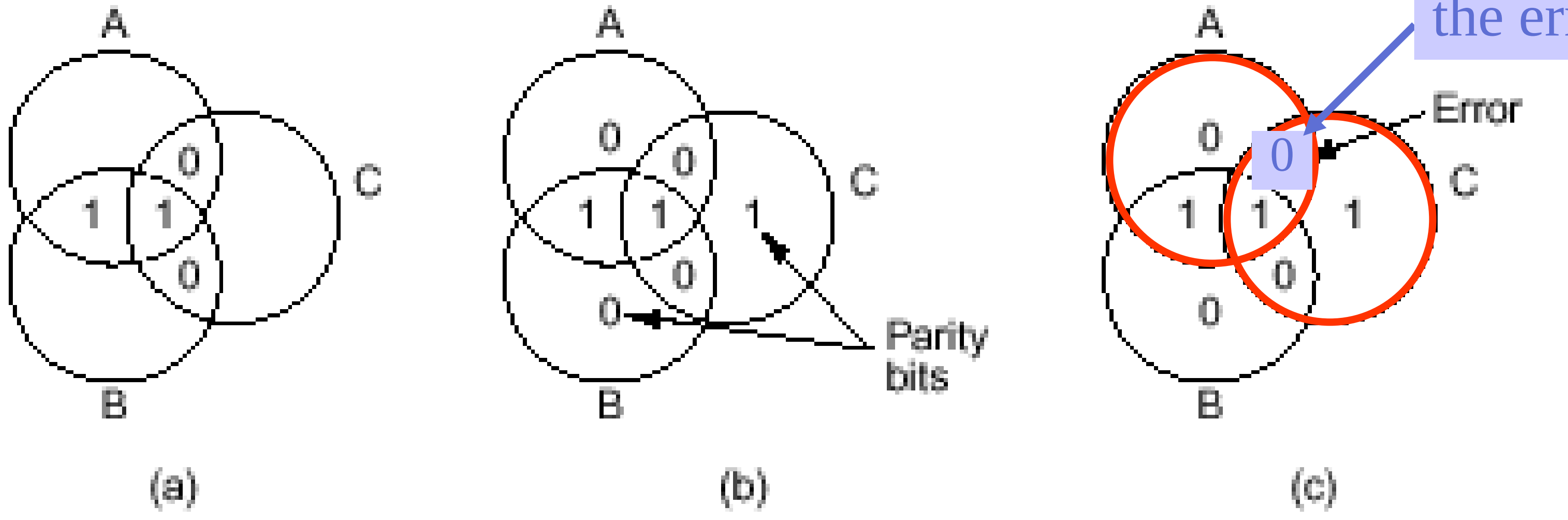


Figure 2-14. (a) Encoding of 1100. (b) Even parity added. (c) Error in *AC*.

[Tane99]

(a) Each data bit (4) belongs to different set of parity sets (3)

(c) Sets A and C detect errors. Only one bit belongs to A and C, but not B. Now we know, which bit was flipped (assume only one bit was flipped).

(b) Need 3 ”extra” parity bits!

What if the error is in parity bit?

Error Correcting Hamming Code

	correct		erroneous
Data:	100 <u>1</u>1<u>00</u>	$\Rightarrow$	1<u>1</u>0 1100
Bit nr:	765 4321		765 4321

(even parity)

1 =	001
2 =	010
3 =	011
4 =	100
5 =	101
6 =	110
7 =	111

Parity bit 1 checks bits 1, 3, 5, 7

Parity bit 2 checks bits 2, 3, 6, 7

Parity bit 4 checks bits 4, 5, 6, 7

Error occurs: bit 6 flips

Parity bit 2 checks bits 2, 3, 6, 7: ERROR

Parity bit 4 checks bits 4, 5, 6, 7: ERROR

2+4 = 6 $\Rightarrow$ bit 6 is bad, <u>correct</u> (flip) it

CRC - Cyclic Redundancy Code

- Integrity check used in data transmissions
- Check sum (16 bits) for large data set (msg?)
  - compute $CRC_S = f(msg) \% 2^{16}$ (last 16 bits)
  - Send msg and CRC_S
  - Recieve msg and CRC_S
  - Compute CRC_R from message and compare it to CRC_S
  - If something wrong, ask msg to be sent again

CRC-CCITT CRCs detect:
All single- and double-bit errors
All errors of an odd number of bits
All error bursts of 16 bits or less
In summary, 99.998% of all errors

Data Integrity Usage Areas

- The more closer to processor, the more important data integrity is
- Internal regs, internal bus, memory bus, other buses (hw)
  - Error correcting Hamming-koodi
  - Extra wires for parity bits
  - Hw-circuits to check parity bits (time to do it!?)
- Networks (sw)
  - CRC with retransmissions
  - If errors occur, there is often many of them
    - Hamming code would not be enough anyway
    - Parity bit alone would not detect (e.g.) 2 bit errors

Replication of devices/systems

- Many memory circuits or disks, with replicated data
- Many CPUs, executing same instructions
- Many systems, executing same programs
  - Voting for each phase: majority wins
  - Complex? Slow?
  - Erroneous system taken out automatically?
    - Replaced by a backup system?
- Same of different types of systms, similar software
  - Same specs, different software teams, same inputs

"Four of the five computers (IBM AP-101) on the space shuttle Columbia ran identical software and compared results with each other before giving the go-ahead to take a specific action. The fifth computer (also IBM AP-101) ran a different version of the software and was used only if the others failed."

http://www.hq.nasa.gov/office/pao/History/computers/contents.html

Cache and Memory

Cache

Register reference time: X
Memory ref. time: 50-200X?
(rough estimate)

- Problem: main memory is pretty far from CPU
- Solution: cache next to CPU
  - Keep copies of recently referenced main memory areas

Cache reference time: ~2-4X?

- Every memory reference is now:
  - Check if referenced data is in cache
  - If data is not in cache, fetch it from main memory
    - CPU just waits idle, hw implementation!
  - Make the reference to data (code or data) from cache
  - (possibly store modified data to main memory)

Discuss

Implementing Memory

- Different technologies to different level of memory
- RAM - Random-Access Semiconductor Memory
  - Give address and read/write signal
  - Any location read/written in the same time
  - Lose power ⇒ lose memory data

random access

volatile memory

All current main memory technologies are ”random access”

Two main Technologies for RAM

- DRAM: dynamic RAM, cheaper, slower, data must be refreshed every (e.g.) every 2 ms
  - Main memory in most systems
  - Implemented with ”leaking” condensators
- SRAM: static RAM, more expensive (~10-20x), faster (~10-50x), takes more space, does not require refreshing
  - Registers in CPU
  - Cache in most systems
  - Memory in high end servers and supercomputers
  - Implemented with similar logic gates as processor

Other Implementation Technologies for RAM

- SDRAM (synchronous dynamic random access memory)
  - Internal buffer, many memory ops in transit
- DDR (double data rate) SDRAM
  - Two data transfers in one clock cycle, one in ascending and the other in descending pulse phase
- …

ROM technologies

- ROM - Read-Only Memory
  - Data sustained without power
  - Can only be read, not written
    - E.g., system initialization code (BIOS)
  - Written at chip manufacture time, Mask-ROM
    - Not in use any more
    - Same technology as CPU design
  - Bad: errors in code cannot be fixed
  - Update: insert new chip onto mother board
  - Random access
  - Usually slower than RAM (~10x)

(non-volatile)

HISTORY

Programmable ROM-memories

- PROM - Programmable ROM
  - Written just once
  - Update: ”burn” data to empty PROM
- EPROM - Erasable PROM
  - Erase all at once, not one word at a time
  - Erase all data with 20 min. UV-radiation in “oven”, load new data to erased EPROM
- EEPROM - Electronically Erasable PROM
  - Can erase electronically individual bytes
- FLASH EEPROM memory
  - Can erase all data at once electronically (while chip in system)
  - Normal voltage, all or one block at a time
  - Faster than EEPROM
- Flash Drive, SSD – Solid State Disk
  - Large memories implemented with flash technology
  - OS sees often (not always) as hard disk

read-mostly memory

HISTORY

HISTORY

CURRENT

SSD – Solid State Disk

- Packaged Flash-memory (e.g.)
- Limited number or writes per block (e.g., 100 000)
  - Written physical memory blocks are rotated
  - Spare blocks ready to be taken into use
  - File cache in main memory may reduce number of writes
- Use time: 5-10v? (not forever!)
- Many technologies
  - nor: read/write one word at a time, slower
  - nand: read/write one block at a time, faster

SSD vs. Hard Disk (HDD)

- I/O per sec: SSD maybe 100x faster
- Access time: SSD maybe 10-100x faster
- Capacity: HDD maybe 4x
- SSD stands vibration, does not wear out, talks less power, takes less space
- Hinta: SSD ehkä 8x (2016)
  - HDD 2 TB, 70e
  - SSD 1 TB, 280e
- Eventually SSD will probably replace HDD

Memory Hierarchy

- Flash?
- SSD?
- File server?
- Cloud?

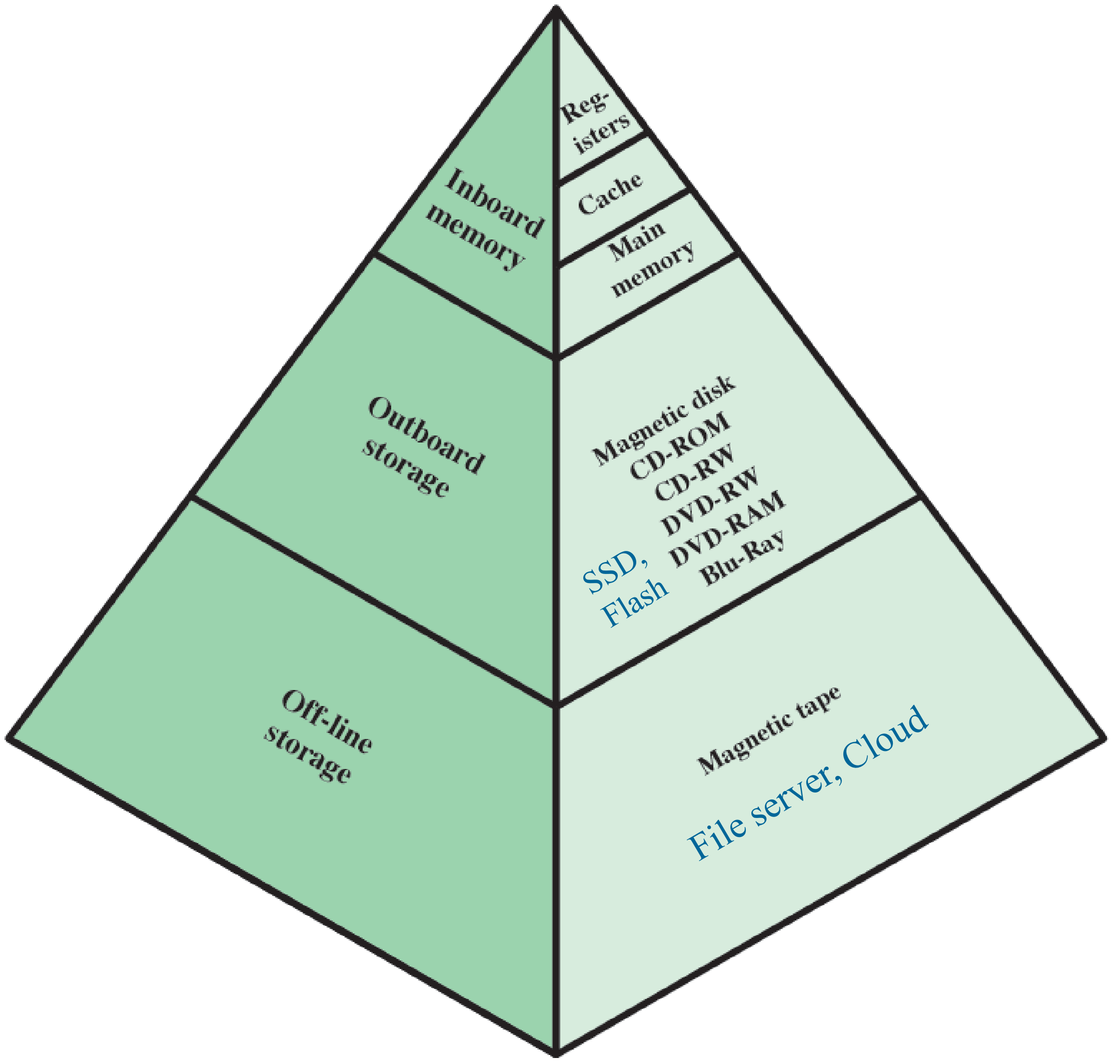


Figure 4.1 The Memory Hierarchy

-- End --

- Acoustic mercury tube
  - Piezielectric quartz crystal changed electric current to acoustic signal (and vice versa)
  - 1000 bits per 1.45m tube
  - Read: wait until wanted data is at the end of tube, read it, and write it back to the other end of tube
  - W. Shockley & J.P. Eckert, 1946
  - M. Wilkes, EDSAC – Electronic Delay Storage Automatic Calculator, 1949