FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

HOOFDSTUK 4

GEGEVENSCODERING

Helga Naessens

UNIVERSITEIT
GENT

Gegevenscodering

- intern geheugen bevat programmaopdrachten en **te verwerken / verwerkte gegevens**
- hoe worden gegevens gecodeerd in intern geheugen?
- welke binaire getallen worden gebruikt voor welk soort gegevens?
  - getallen, tekst, …
    - een groep van tekens (bv. een getal)
    - elk teken afzonderlijk (bv. tekst)
  - beeld, geluid

Inhoud

- numerieke informatie
  - **gehele getallen**
  - rationale getallen
  - decimale getallen
- alfanumerieke informatie (tekst)
- voorstelling en opslag van beeld en geluid
- compressie van gegevens + bestandsformaten

Gehele getallen: fixed point codering

- gecodeerd in een **vast aantal** bits
- **cyclische** eigenschap:
  - voorbeeld: 3 decimale cijfers

    van 000 tot 999 en nadien 000
- hoe meer bits hoe groter de getallen kunnen zijn
- aantal mogelijke getallen bij N bits: 2^N
- voor de eenvoud gebruiken we in de volgende voorbeelden

  3 bits $\Rightarrow$ 8 mogelijkheden

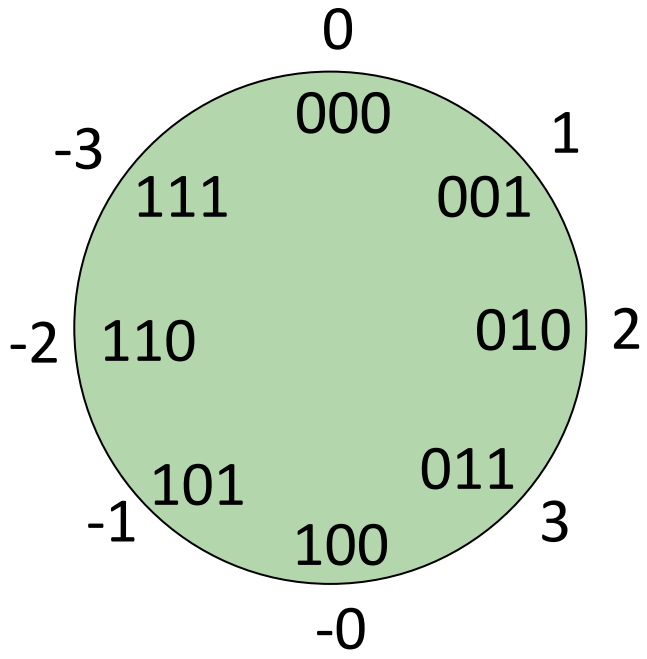
fixed point: absolute value & sign

- Linkse bit = tekenbit:

  0 = positief

  1 = negatief

- Overige bits: binaire voorstelling van de absolute waarde
- Nadelen:
  - twee voorstellingen van nul
  - twee discontinuïteiten



fixed point: binary offset (excess)

- code = getal + Δ , met $\Delta = 2^{N-1}$

  $\Rightarrow$ verschuiving over Δ

- Voordelen:

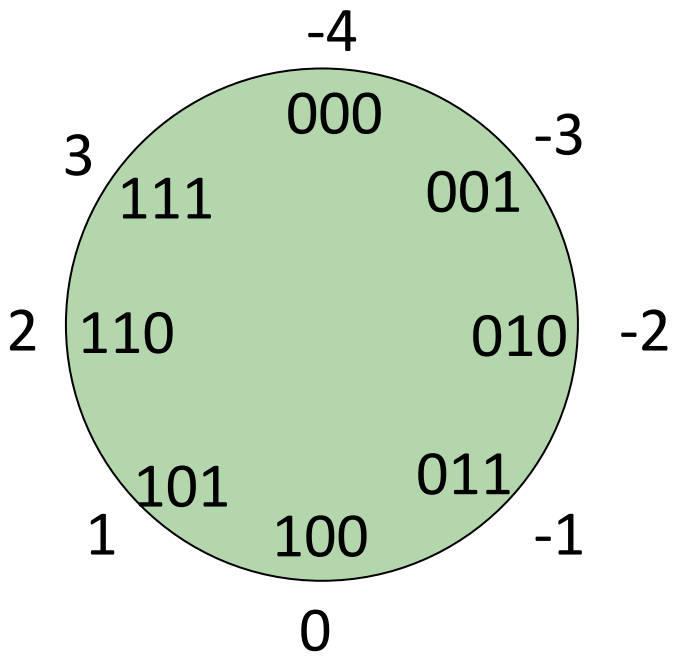
  + één voorstelling van nul

  + één discontinuïteit

- Nadeel:

  - voorstelling voor nul

    ≠ alle bits 0



fixed point: 2-complementcodering

- Linkse bit = tekenbit:

  0 = positief , 1 = negatief

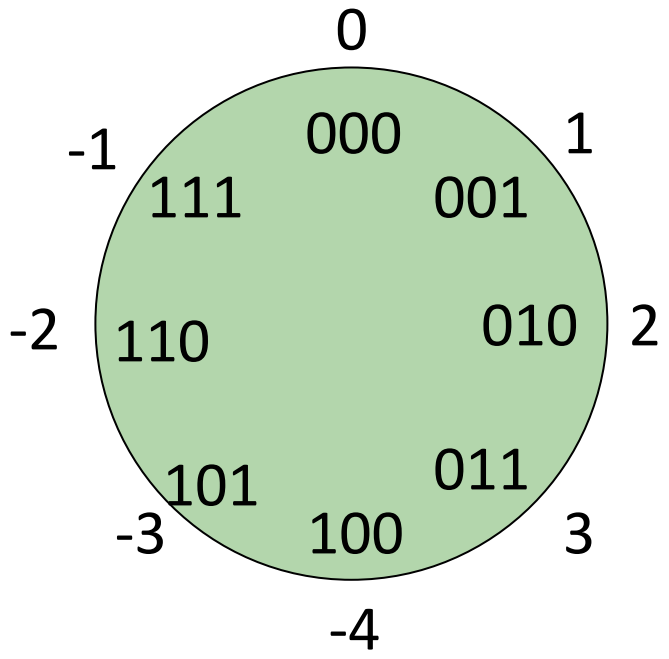
- Getallen ≥ 0: binaire voorstelling
- Getallen < 0: 2-complement

  van de absolute waarde

- Voordelen:

  + één voorstelling van nul

  + één discontinuïteit



2-complementcodering: + en -

- -1 + 3 = ?

  111 + 011 = 1010 → 2

- 2 – 3 = 2 + (– 3) = ?

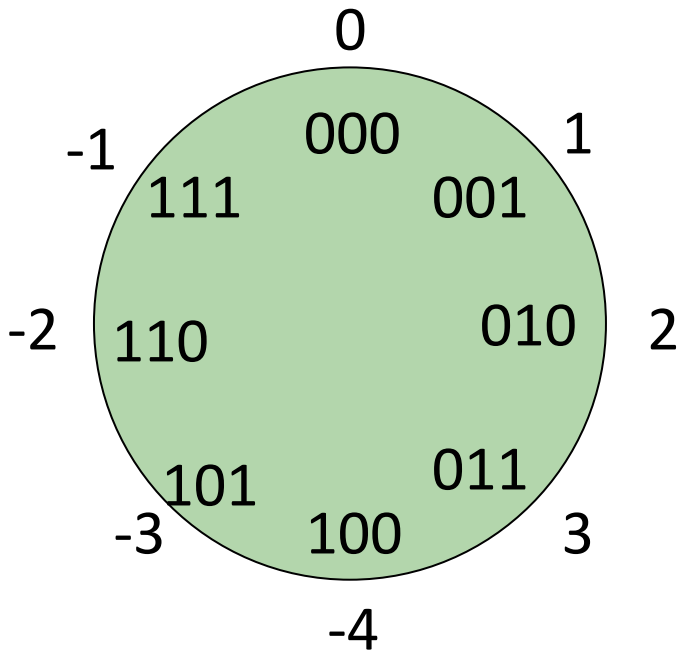
  010 + 101 = 111 → -1

- 3 – 2 = ?

  011 + 110 = 1001 → 1

- 3 + 2 = ?

  011 + 010 = 101 → -3 (geen foutmelding)



2-complementcodering: gevolgen

- bereik in n bits: $[\ -2^{n-1},\ 2^{n-1} - 1\ ]$
- meestal 16 of 32 bits
  - bereik 16-bits: [–32 768, 32 767]
  - bereik 32-bits: [–2 147 783 648, 2 147 483 647]
- let op: programma geeft geen foutmelding bij overschrijding van het bereik

  2 147 483 647 + 1 $\rightarrow$ – 2 147 783 648

- Python: willekeurige precisie
- https://www.youtube.com/watch?v=Z3mswCN2FJs

Inhoud

- numerieke informatie
  - gehele getallen
  - **rationale getallen**
  - decimale getallen
- alfanumerieke informatie (tekst)
- voorstelling en opslag van beeld en geluid
- compressie van gegevens + bestandsformaten

Rationale getallen: welke codering

- codering van gehele getallen: fixed point codering

  $\Rightarrow$ punt op een vaste plaats

- is fixed point codering bruikbaar voor rationale getallen?
  - ja, maar … plaats van punt afspreken
  - nadeel: sterk uiteenlopende waarden zijn niet mogelijk (bewijs volgt)

Floating point codering

- vlottendekommavoorstelling
- vergelijk met wetenschappelijke notatie rekentoestel
- getal = fractie x grondtal^{exponent}
  - fractie en exponent worden opgeslagen
  - grondtal impliciet
  - $0 \leq \text{fractie} < 1$
  - teken bij fractie en exponent: + of –

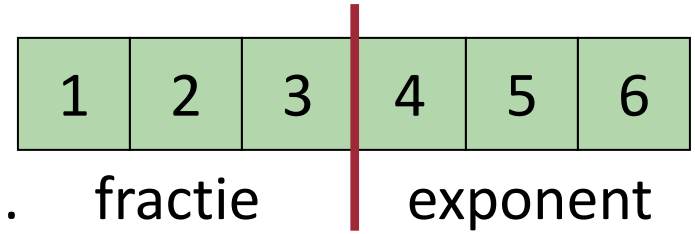
Bijvoorbeeld: $12.34 = 0.1234 \times 10^2$

Floating point codering

- **decimaal voorbeeld**
- algemene kenmerken
- gevolgen
- bestaande coderingen

Decimaal voorbeeld: afspraken

- onderstel: 6 decimale cijfers beschikbaar
  - 3 cijfers voor de fractie
  - 3 cijfers voor de exponent
- alleen positieve getallen
- alleen positieve exponenten
- grondtal 10

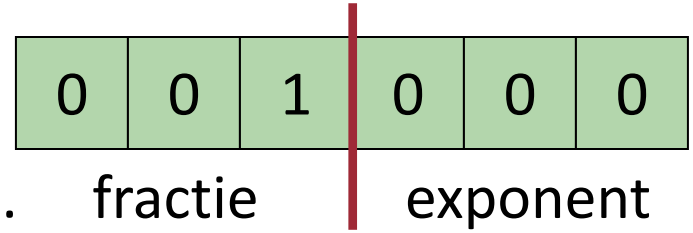


$\Rightarrow 0.123 \times 10^{456}$

Decimaal voorbeeld: gevolgen

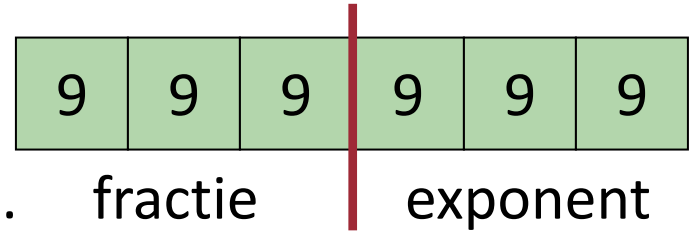
- kleinste getal (≠ 0):

  $0.001 \times 10^{0} = 0.001 = 10^{-3}$



- grootste getal:

  $0.999 \times 10^{999} \approx 10^{999}$



Dynamisch bereik

- dynamisch bereik van de voorstelling =

  grootste / kleinste (≠ 0)

- bij floating point volgens vorige afspraken:

  $10^{999} / 10^{-3} = 10^{1002}$

- bij fixed point codering met 6 cijfers:

  $999999 / 1 \approx 10^{6}$

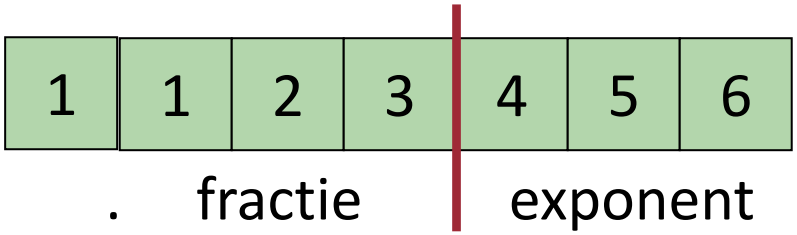
$\Rightarrow$ groter bereik bij floating point

Floating point codering

- decimaal voorbeeld
- algemene kenmerken
  - **tekenbit**
  - genormaliseerde mantisse
  - exponent in excess-code
- gevolgen
- bestaande coderingen

Tekenbit

- extra cijfer nodig om teken van fractie voor te stellen
- meestal meest linkse bit
- 0 voor positief, 1 voor negatief
- decimaal voorbeeld:



$\Rightarrow -0.123 \times 10^{456}$

Floating point codering

- decimaal voorbeeld
- algemene kenmerken
  - tekenbit
  - **genormaliseerde mantisse**
  - exponent in excess-code
- gevolgen
- bestaande coderingen

Genormaliseerde mantisse

- mantisse zijn cijfers na de komma

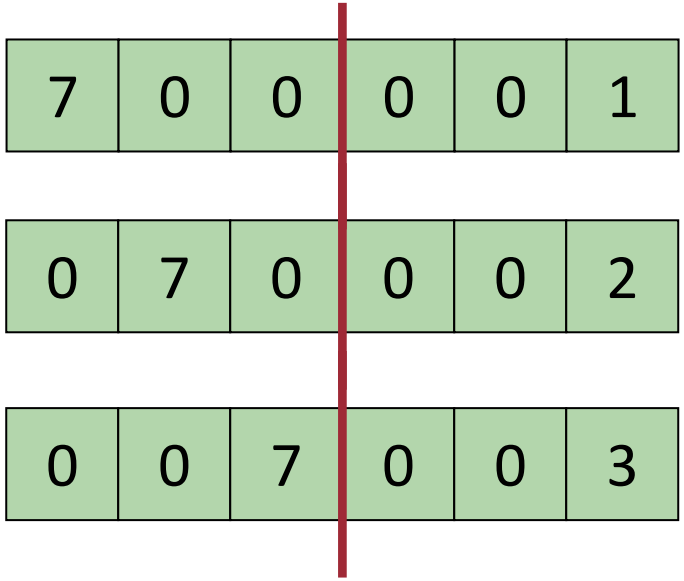
  $\Rightarrow$ fractie = 0.mantisse

- exponentiële notatie is niet eenduidig

  7 = ?

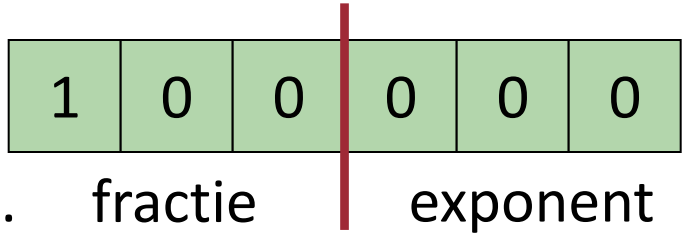
  - 0.7×10^{1}
  - 0.07×10^{2}
  - 0.007×10^{3}

- beste notatie?



Genormaliseerde mantisse (2)

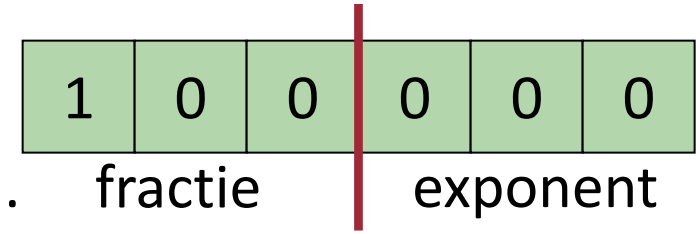
- genormaliseerd: meest beduidende (= linkse) cijfer van mantisse is niet nul



- Uitzondering: nul
  - ➢ exponent = mantisse = 0

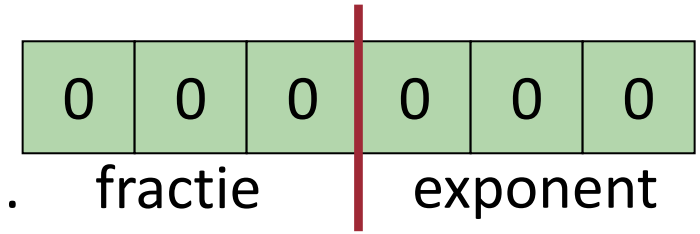
Decimaal voorbeeld

- kleinste getal (≠ nul)



$0.100 \times 10^{0} = 0.100 = 10^{-1}$

- nul



Floating point codering

- decimaal voorbeeld
- algemene kenmerken
  - tekenbit
  - genormaliseerde mantisse
  - **exponent in excess-code (cfr binary offset)**
- gevolgen
- bestaande coderingen

Exponent in excess-code

- doel: geen extra cijfer voorzien voor negatieve exponenten
- excess-code is binary offset codering
- verschuiving over de helft van de waarden
- verschuiving **$\Delta = \frac{1}{2}\ \text{grondtal}^n$**

  n = aantal cijfers exponentvoorstelling
- voorstelling (codering) van exponent =

  waarde van exponent + Δ (=offset, excess)

Decimaal voorbeeld

- exponent in 3 cijfers
- excess $\Delta = ½\ 10^3 = 500$
- exponentcodering = waarde exponent + 500

  $\Rightarrow$ 000 ≤ exponentvoorstelling ≤ 999

  $\Rightarrow$ –500 ≤ exponentwaarde ≤ 499

- <u>voorbeeld</u>

  2 voorgesteld door 502

  -3 voorgesteld door 497

Floating point codering

- decimaal voorbeeld
- algemene kenmerken
- **gevolgen**
  - **voor de grenzen van de voor te stellen getallen**
  - voor de nauwkeurigheid
- bestaande coderingen

Grenzen: minimum

- kleinste positief getal (≠ nul)

  $\text{min} = 0.M_{min} \times \text{grondtal}^{E_{min}}$

  - M_{min}: kleinste waarde mantisse
  - E_{min}: kleinste waarde exponent

- decimaal voorbeeld
  - M_{min}: 1
  - E_{min}: codering 000 $\rightarrow$ waarde = -500
  - $\text{min} = 0.1 \times 10^{-500}$

Grenzen: maximum

- grootste positief getal

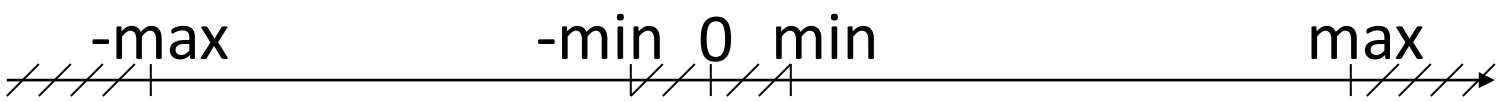
  $max = 0.M_{max} \times grondtal^{E_{max}}$

  - M_{max}: grootste waarde mantisse
  - E_{max}: grootste waarde exponent

- decimaal voorbeeld
  - M_{max}: 999
  - E_{max}: codering 999 $\rightarrow$ waarde = 499
  - $max = 0.999 \times 10^{499}$

Grenzen: conclusie

- niet alle getallen kunnen voorgesteld worden



overflow **underflow** **overflow**

- **overflow**: resultaat van een berekening heeft een te grote absolute waarde
- **underflow**: resultaat van een berekening heeft een te kleine absolute waarde

  $\Rightarrow$ nul is een uitzondering
- genereren foutmeldingen in programma

Grenzen

- worden vooral bepaald door grenzen van exponent
- bepaald door E_{min} en E_{max}
- want binair geldt
  - $0.M_{min} = (0.1)_2 = (0.5)_{10}$
  - $0.M_{max} = (0.111111\ldots11)_2 \approx (1)_{10}$

$\Rightarrow$ beperkte invloed van mantisse

Floating point codering

- decimaal voorbeeld
- algemene kenmerken
- **gevolgen**
  - voor de grenzen van de voor te stellen getallen
  - **voor de nauwkeurigheid**
- bestaande coderingen

Nauwkeurigheid

- hoeveel beduidende cijfers worden er bewaard?
- afhankelijk van het aantal cijfers (bits) van de mantisse
- decimale nauwkeurigheid d:

$$\mathbf{d = m \log_{10} 2} \qquad (\Leftarrow 10^{-d} = 2^{-m})$$

(m = aantal bits voorzien voor de mantisse)

Floating point codering: overzicht

- getal = **fractie** x grondtal$^{\textbf{exponent}}$
  - fractie ($0 \leq \text{fractie} < 1$) en exponent opgeslagen
  - teken van het getal apart opgeslagen in 1 cijfer (bit)
- **mantisse** zijn cijfers na de komma (fractie = 0.mantisse)
  - mantisse genormaliseerd → eerste cijfer beduidend
- **exponent in excess-code**: code = waarde + Δ
- gevolgen
  - grenzen bepaald door exponent (overflow, underflow)
  - dec. nauwkeurigheid d bepaald door aantal mantissebits m

    $d = m \log_{10} 2$

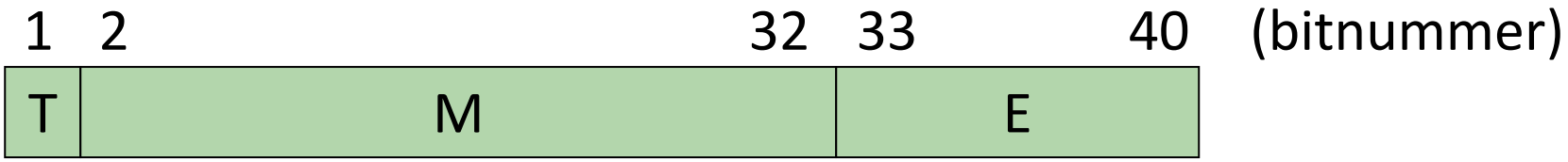
Floating point codering

- decimaal voorbeeld
- algemene kenmerken
- gevolgen
- **bestaande coderingen**

Floating point: bestaande coderingen

- Opmerking:
  - niet van buiten te kennen
  - wel gevolgen kunnen berekenen indien afspraken gekend!
- **APPLE-II (eenvoudige codering)**
- IEEE-formaat
  - enkele precisie (32 bits)
  - dubbele precisie (64 bits)

APPLE-II: teken en mantisse

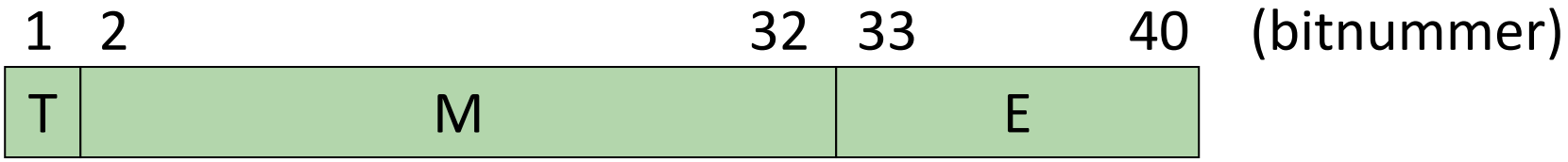


- bit 1: tekenbit

  0: positieve getallen 1: negatieve getallen

- bit 2 .. 32: mantisse
  - $0.M_{min} = (0.100\ldots0)_2 = 0.5$
  - $0.M_{max} = (0.111\ldots1)_2 \approx 1$

APPLE-II: exponent

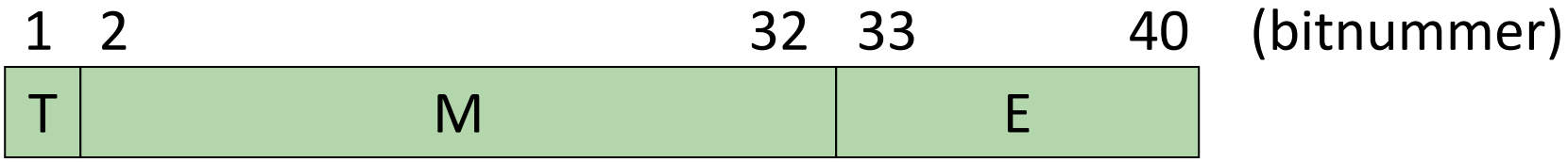


- bit 33 .. 40: exponent
  - 8 bits → excess = $2^8/2 = 2^7 = 128$
  - $E_{min} = (00000000)_2 = 0$

    voorgestelde waarde = code - $\Delta = 0 - 128 = -128$
  - $E_{max} = (11111111)_2 = 255$

    voorgestelde waarde: $255 - 128 = 127$

APPLE-II: gevolgen



- grenzen
  - min = $0.5 \times 2^{-128} = 2^{-129} \approx 1.47 \times 10^{-39}$
  - max $\approx 1 \times 2^{127} \approx 1.7 \times 10^{38}$
- decimale nauwkeurigheid: $d = 31 \log_{10} 2 \approx 9.33$

  $\rightarrow$ 9 beduidende cijfers

APPLE-II: voorbeeld

- hoe wordt -23,75 gecodeerd?
- werkwijze:
  - omzetten naar binair
  - mantisse normaliseren
  - mantisse en exponent coderen
  - tekenbit bepalen
- omzetten naar binair: $23,75 = (?)_2$

  $23,75 = (10111.11)_2$
- mantisse normaliseren:

  $10111.11 = 0.1011111 \times 2^5$

APPLE-II: voorbeeld

$$23{,}75 = 0.1011111 \times 2^5$$

- mantisse en exponent coderen
  - M = 1011 1110 0000 0000 0000 0000 0000 000

    wat als meer beduidende bits? afronden (zie verder)
  - E (code) = waarde van exponent + Δ = 5 + 128 =

    101 + 1000 0000 = 1000 0101
- tekenbit bepalen: T = 1

Resultaat

1 1011 1110 0000 0000 0000 0000 0000 000 1000 0101

Floating point: bestaande coderingen

- APPLE-II (eenvoudige codering)
- **IEEE-formaat**
  - **enkele precisie (32 bits)**
  - **dubbele precisie (64 bits)**

Floating point: IEEE-formaat

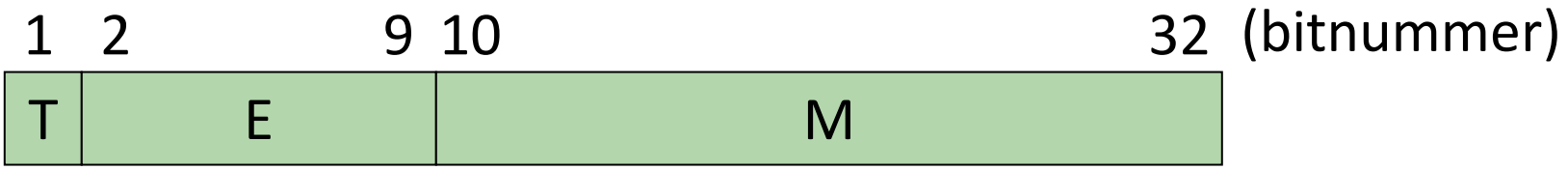
- Institute of Electrical and Electronics Engineers
- twee formaten
  - 32-bits codering (enkele precisie)
  - 64-bits codering (dubbele precisie)
- 2 wijzigingen t.o.v. vorig model: mantisse en exponent
- genormaliseerde mantisse
  - mantisse begint steeds met 1
  - waarom die 1 opslaan in de codering?
  - die eerste 1 niet opslaan $\rightarrow$ 1 bit extra nauwkeurigheid
- $(1101)_2 = (0.1101)_2 \times 2^4 = (1.101)_2 \times 2^3$
- wat betekent M=0 en E=0?? nul of 1.0×2^{-N}?

IEEE-formaat: exponent

- onduidelijkheid wegwerken
- mogelijke exponentcoderingen: 0..E_{max}
- 0 en E_{max} hebben speciale betekenis
  - E = 0 en M = 0: stelt nul voor
  - E = 0 en M ≠ 0: niet genormaliseerde mantisse
  - E = E_{max} en M = 0: stelt ∞ voor (geen fout bij overflow)
  - E = E_{max} en M ≠ 0: geen getal (NAN, not a number)
- gevolg: mogelijke exponentcoderingen 1..(E_{max}-1)
  - excess is 1 minder dan normaal
  - excess: **$\Delta = ½ \text{ grondtal}^s - 1$**

    (s = aantal cijfers exponentvoorstelling)

IEEE-formaat: enkele precisie



- bit 1: tekenbit

  0: positieve getallen 1: negatieve getallen

- bit 10 .. 32: mantisse (23 bits)
  - $1.M_{min} = (1.000\ldots0)_2 = 1$
  - $1.M_{max} = (1.111\ldots1)_2 \approx 2$

IEEE-formaat: enkele precisie

- bit 2 .. 9: exponent (8 bits)
  - 8 bits → excess = $(2^8-2)/2 = 2^7 - 1 = 127$
  - $E_{min} = (00000001)_2 = 1$

    voorgestelde waarde = code - $\Delta = 1 - 127 = -126$
  - $E_{max} = (11111110)_2 = 254$

    voorgestelde waarde: $254 - 127 = 127$

IEEE-formaat: gevolgen

- grenzen
  - min = $1 \times 2^{-126} \approx 1.17 \times 10^{-38}$
  - max $\approx 2 \times 2^{127} \approx 3.4 \times 10^{38}$
- decimale nauwkeurigheid: $d = (23+1) \log_{10} 2 \approx 7.22$

  $\rightarrow$ 7 beduidende cijfers

IEEE-formaat: niet genormaliseerde getallen

- E = 0
- M ≠ 0 (niet genormaliseerd)
  - kleinste mantisse: 00..01
  - $0.M_{kl} = (0.000\ldots01)_2 = 2^{-23}$

$\Rightarrow$ verlaagde ondergrens: $2^{-23} \times 2^{-126} \approx 1.4 \times 10^{-45}$

IEEE-formaat: voorbeeld

Hoe wordt −0.035 gecodeerd?

- $0.035 = (0.0000\ 1000\ 1111\ 0101\ 1100\ 0010\ 1000\ 1111\ldots)_2$

  $= (1.000\ 1111\ 0101\ 1100\ 0010\ 1000\ 1111\ \ldots)_2 \times 2^{-5}$

- M = 000 1111 0101 1100 0010 1001

  (afronden: eerste niet genoteerde bit=1 $\rightarrow$ 1 bijtellen)

- E (code=waarde+Δ) = -5 + 127 = 122 = $(0111\ 1010)_2$

- T = 1

IEEE-formaat: dubbele precisie

- enkele precisie: 7 beduidende cijfers
- hoe dubbele precisie bereiken?
- aantal bits uitbreiden van mantisse
- aantal verdubbelen?
- neen, ook 3 bits extra voor exponent
- gevolg: ook groter bereik

IEEE-formaat: dubbele precisie

- bit 1: tekenbit

  0: positieve getallen 1: negatieve getallen

- bit 13 .. 64: mantisse (52 bits)
  - $1.M_{min} = (1.000\ldots0)_2 = 1$
  - $1.M_{max} = (1.111\ldots1)_2 \approx 2$

IEEE-formaat: dubbele precisie

- bit 2 .. 12: exponent (11 bits)
  - 11 bits → excess = $(2^{11}-2)/2 = 2^{10} - 1 = 1023$
  - $E_{min} = (00000000001)_2 = 1$

    voorgestelde waarde = code - $\Delta = 1 - 1023 = -1022$
  - $E_{max} = (11111111110)_2 = 2046$

    voorgestelde waarde: $2046 - 1023 = 1023$

IEEE-formaat: gevolgen

- grenzen
  - min = $1 \times 2^{-1022} \approx 1.23 \times 10^{-308}$
  - max $\approx 2 \times 2^{1023} \approx 1{,}8 \times 10^{308}$
- decimale nauwkeurigheid: $d = (52+1) \log_{10} 2 \approx 15.95$

  $\rightarrow$ 15 beduidende cijfers

IEEE-formaat: niet genormalisserde getallen

- E = 0
- M ≠ 0 (niet genormaliseerd)
  - kleinste mantisse: 00..01
  - $0.M_{kl} = (0.000\ldots01)_2 = 2^{-52}$

$\Rightarrow$ verlaagde ondergrens: $2^{-52} \times 2^{-1022} \approx 4.94 \times 10^{-324}$

Inhoud

- numerieke informatie
  - gehele getallen
  - rationale getallen
  - **decimale getallen**
- alfanumerieke informatie (tekst)
- voorstelling en opslag van beeld en geluid
- compressie van gegevens + bestandsformaten

Decimale getallen

- waar?
  - administratieve toepassingen
  - weinig (eenvoudige) berekeningen
  - volledige nauwkeurigheid vereist
- doel: geen afrondingsfouten
- dit vereist het opslaan van
  - elk cijfer afzonderlijk
  - teken
  - komma
  - einde getal
- binaire code van 4 bits: BCD (binary coded decimal)

Inhoud

- numerieke informatie
  - gehele getallen
  - rationale getallen
  - decimale getallen
- **alfanumerieke informatie (tekst)**
- voorstelling en opslag van beeld en geluid
- compressie van gegevens + bestandsformaten

Alfanumerieke informatie

- elk teken afzonderlijk coderen in een eigen bitpatroon
- soorten tekens
  - afdrukbaar
  - niet-afdrukbaar (controletekens)
- codering
  - ASCII (7 + 1 bits)
  - UNICODE (16 bits)

ASCII

- American Standard Code for Information Interchange
- 7 bits $\rightarrow$ 128 tekens
- byte: 8 bits
  - 0000 0000 $\rightarrow$ 0111 1111: ASCII
  - 1000 0000 $\rightarrow$ 1111 1111: geen standaard, afhankelijk van operating system
- overzicht pagina 50 in cursus
  - controletekens (niet-afdrukbare tekens)
  - andere tekens: hoofdletters, kleine letters, cijfers, …

Inhoud

- numerieke informatie
- alfanumerieke informatie (tekst)
- **voorstelling en opslag van**
  - **beelden**
  - geluid
- compressie van gegevens + bestandsformaten

Voorstelling van beelden

- twee manieren om beelden op te slaan
  - als bitmapafbeelding
  - als vectortekening
- software bepaalt hoe beelden bewaard worden
  - vb. CorelDraw vs. Corel PhotoPaint
  - vb. Microsoft Visio vs. Paint

Beelden voorgesteld als bitmap

- Principe:

  beeld = verzameling punten (pixels, picture element)

- **elke pixel wordt gecodeerd**
  - code is bepaald door manier waarop pixel zich voordoet: kleur, helderheid
  - voorstelling van beeld = verzameling gecodeerde pixels $\rightarrow$ bitmap

- veel displaytoestellen (printer, scherm) werken ook met dit pixelconcept $\rightarrow$ populaire techniek

Codering van pixels

- afhankelijk van toepassing
- **zwart-witbeeld**: 1 pixel = 1 bit (0 of 1)
- **beeld met grijstinten** (vb. zwart-wit foto)
  - 1 pixel = meerdere bits (vb. meestal 8)
  - laat toe 256 verschillende grijstinten voor te stellen
- **kleurenbeeld**
  - meer complex systeem
  - twee benaderingen veel gebruikt
    - codering van helderheid + 2 kleurcomponenten
    - RGB codering

Codering van kleurenpixels

- code voor helderheid + 2 kleurcomponenten:

  populair om TV beeld (kleur) op ZW toestel te tonen

- **RGB-codering**:
  - 3 componenten worden apart gecodeerd:

    Rood, Groen, Blauw
  - vaak 1 byte per component
  - omvang: 3 bytes per pixel (24 bits)

- andere methodes: geïndexeerde kleuren
  - pixelwaarde duidt de kleur uit tabel van kleuren
  - 4 of 8 bits per pixel

Bitmap: bedenkingen

- beeldformaat niet zonder meer aan te passen
- bij **verkleinen**: minder pixels

  → welke pixels van origineel beeld weglaten?
- bij **vergroten** = inzoomen op beeld
  - aantal pixels neemt toe (800x600 → 1024x768)
  - originele pixels vergroten
    - extra pixels per origineel pixel
    - korrelig effect
  - gebruikt bij digitale zoom fototoestel
- nadeel weggewerkt bij vectortekening

Beeld als vectortekening

- beeld = verzameling geometrische structuren

  vb. lijnen en krommen
- codering structuren d.m.v. techniek uit analytische meetkunde
- toestel dat beeld moet tonen beslist over voorstelling van deelfiguren
- o.a. gebruikt bij
  - schaalbare lettertypes (vb. TrueType, Postscript)
  - software voor CAD (computer aided design)

Inhoud

- numerieke informatie
- alfanumerieke informatie (tekst)
- **voorstelling en opslag van**
  - beelden
  - **geluid**
- compressie van gegevens + bestandsformaten

Voorstelling van geluid: sampling

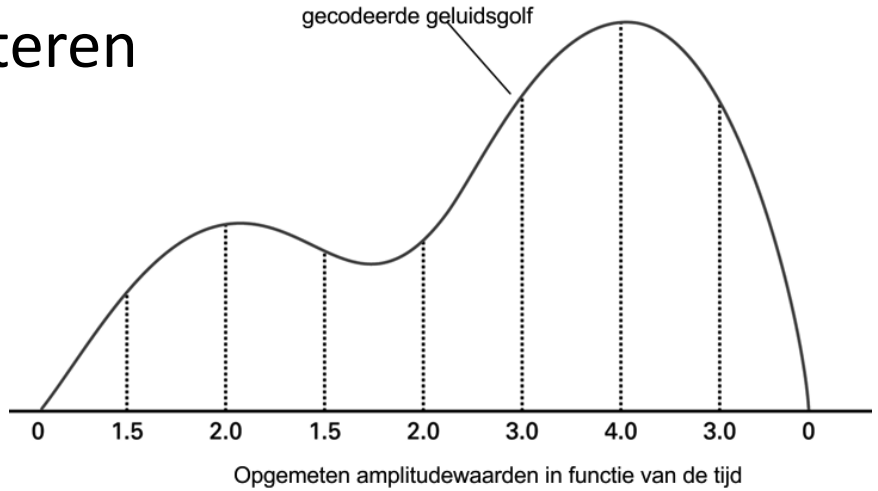
- is meest algemene manier van voorstellen
- amplitude geluidsgolf bemonsteren op regelmatige tijdsintervallen

  → waarden opslaan
- frequentie?
  - telefoon: 8000 samples/sec
  - muziek CD's: 44100 Hz

    → elke waarde 16 bits (32 stereo)

    → 1 sec muziek in stereo

    = 1,4 miljoen bits (44100x32) = 172 KB



Voorstelling van geluid: MIDI

- Musical Instrument Digital Interface
- gebruikt bij muzieksynthesizers
- muziek wordt niet gecodeerd
- opslag van instructies voor productie van muziek:
  - welk instrument
  - welke tijdsduur
  - klarinet speelt 2 seconden een RE $\rightarrow$ 3 bytes
- cf. opslag van muziekpartituur

Inhoud

- numerieke informatie
- alfanumerieke informatie (tekst)
- voorstelling en opslag van
  - beelden
  - geluid
- **compressie van gegevens + bestandsformaten**

Compressie van gegevens

- veel geheugenruimte vereist voor beeld en geluid
- nood aan gegevenscompressie: opslagruimte wordt gereduceerd (= minder bits)
- 2 technieken
  - zonder verlies aan informatie
  - met beperkt verlies van informatie
    - alleen aanvaardbaar in sommige situaties
    - bij geluid en beeld zijn beperkt aantal fouten niet waarneembaar

Compressietechnieken: overzicht

- run-length encoding
- frequency-dependent encoding
- relative encoding – differential encoding
- dictionary encoding
- adaptive dictionary encoding (dynamic)

Run-length encoding

- interessant als informatie lange sequenties van dezelfde waarde bevat
- elke waarde die meerdere keren na elkaar voorkomt:

  waarde wordt genoteerd + aantal keer dat ze voorkomt
- bijvoorbeeld:
  - 118x ‘1’ gevolgd door 294x ‘0’ gevolgd door 58x ‘1’
  - neemt minder plaats dan patroon 470 bits

Frequency-dependent encoding

- elk item wordt gecodeerd met verschillende lengte (≠ ASCII en UNICODE)
- lengte van code van item is omgekeerd evenredig met frequentie van voorkomen van item
- in Nederlandse tekst: e, n, a frequenter dan x, y, q
  → code voor e, n, a korter dan code voor x, y, q
  → plaats bespaard
- nadeel: code van variabele lengte
- worden Huffman-codes genoemd

Relative encoding

- ook *differential encoding* genoemd
- als item in gegevensstroom weinig verschilt van vorig item

  bv. opeenvolgende beeldframes in film
- verschillen tussen items worden opgeslagen

  niet items zelf
- kan zonder en met gegevensverlies: noteer verschillen
  - exact
  - bij benadering

Dictionary encoding

- boodschap bestaat uit bouwblokken die opgeslagen zijn in "woordenboek"
- code is verwijzing naar plaats in woordenboek
- verlies bij compressie?

  enkel als woordenboek slechts benadering van dataelementen bevat (beeldcompressie)
- vb. 25000 woorden in boek $\rightarrow$ plaats tussen 0 en 24999

  $\rightarrow$ 15 bits per woord volstaan
  - ➢ woord van 6 letters in ASCII: 6x8=48 bits
  - ➢ nu: 15 bits

Adaptive dictionary encoding

- ook wel *dynamic* genoemd
- woordenboek wordt bijgewerkt tijdens coderen
- voorbeeld: **LZW methode** (Lempel-Ziv-Welsh)
- start eenvoudig woordenboek met elementaire elementen
- bij detectie van groter blok

  $\rightarrow$ blok wordt toegevoegd aan woordenboek
- gebeurt ook bij decompressie

  $\rightarrow$ opslag van eenvoudig woordenboek + samenstelling

  uitgebreide woordenboek

Voorbeeld van LZW codering

- stel vb. codering van *xyx xyx xyx xyx*
- codering: start met woordenboek van 3 elementen:

  x, y + spatie
- code voor xyx : 121 + spatie: 1213
- bakent woord af → xyx opgenomen in WB (pl 4)
- code boodschap: 121343434
- decodering van 121343434:
  - 1213 levert xyx(spatie) op

    → xyx opgenomen in WB
  - boodschap kan volledig gedecodeerd worden

Oefening 1 LZW codering

Dit is een tekst die moet gecomprimeerd worden met LZW-compressie: aba baa baa ab aba

Het niet-uitgebreide woordenboek bevat de tekens a, b en ‘ ’ (spatie) op plaatsen 1, 2 en 3. Wat is de gecomprimeerde tekst? Geef aan hoe je aan het resultaat komt. Geef ook de inhoud van het uitgebreide woordenboek na compressie.

- Woordenboek: 1: a 2:b 3:’ ‘

  4: aba 5: baa 6:ab

- Gecomprimeerde tekst:

  12132113531234

Oefening 1 LZW codering

Dit is een tekst die moet gecomprimeerd worden met LZW-compressie: aba baa baa ab aba

Het niet-uitgebreide woordenboek bevat de tekens a, b en ‘ ’ (spatie) op plaatsen 1, 2 en 3. Wat is de gecomprimeerde tekst? Geef aan hoe je aan het resultaat komt. Geef ook de inhoud van het uitgebreide woordenboek na compressie.

- Woordenboek: 1: a 2:b 3:’ ‘

  4: aba 5: baa 6:ab

- Gecomprimeerde tekst:

  12132113531234

Opslag van beelden

- bij opslag of transfer: omvang van cruciaal belang
- indien geen compressie: bestandsgrootte (in bytes) = breedte (in pixels) * hoogte (in pixels) * aantal bytes per pixel
- bv. BMP (bitmap, o.a. in Windows)
  - afhankelijk van kwaliteit van afbeelding is 1 pixel:
    1 bit (2 kleuren), 8 bits (256 kleuren),
    true-color (RGB, 24 bits, 16.777.216 kleuren)
  - afbeelding 1280x1024 pixels in true-color-BMP : 4 MB

➔ **compressie gewenst**

Compressie van beelden

- Veelgebruikte formaten:
  - GIF
  - PNG
  - JPEG
  - TIFF
- zijn extensies van bestandsnamen

Compressie van beelden: GIF

- Graphic Interchange Format
- dictionary encoding system: GIF gebruikt LZW-technieken
- code voor pixel : NIET in 3 bytes (geen 16.777.216 kleuren), wel in 8 bits (256 kleuren)
  - ➢ kleuren gekozen op basis van het oorspronkelijk beeld
    → palet (dit palet fungeert als woordenboek)
  - ➢ enkel opslagwinst als oorspronkelijk TrueColor RGB
    → doch verlieslatend (kleuren worden benaderd)
- GIF laat transparante achtergrond en animatie toe en is **geschikt voor eenvoudige tekeningen** (minder voor foto's)

Voorbeeld BMP versus GIF

BMP-formaat

302 KB

GIF-formaat

59 KB

Compressie van beelden: PNG

- PNG (portable network graphics)
- vervanger van GIF
- niet gepatenteerd
- laat 24 bits per pixel toe
- geeft betere resultaten op grote beelden
- laat evenwel geen animatie toe (GIF wel)
- veel gebruikt op WWW

Voorbeeld BMP versus PNG

BMP-formaat

302 KB

PNG-formaat

299 KB

Compressie van beelden: JPEG

- Joint Photographic Experts Group
- veel gebruikt in fotoindustrie (vb. digitale camera)
- minder geschikt voor lijntekeningen
- kent verschillende compressieformaten,
  zelfs zonder verlies (wel nauwelijks compressie)
- meest gebruikt is de baseline standaard
  - compressie in verschillende stappen
  - “misbruikt” beperkingen menselijk oog
    (oog is gevoeliger voor veranderingen in helderheid
    eerder dan in kleur)

Compressiestappen JPEG

- beeld = componenten van helderheid + van kleur
- stap 1: gemiddelde kleurcomponent van 2x2pixels
  $\rightarrow$ kleurinfo/4 $\rightarrow$ geen vermindering van zichtbare kwaliteit (helderheid onaangeroerd)
- stap 2: beeld verdelen in blokken van 8x8 pixels en elk blok vervangen door ander blok dat weergeeft hoe pixels zich onderling verhouden + waarden beneden een ondergrens worden vervangen door nullen (te zwak voor oog)
- stap 3: klassieke compressietechnieken $\rightarrow$ 10:1

Voorbeeld BMP versus JPG

BMP-formaat

302 KB

JPG-formaat

35 KB

Compressie van beelden: TIFF

- Tagged Image File Format
- niet populair omwille van compressie-eigenschappen
- wel standaardformaat om foto's op te slaan +
  extra info: datum, tijd, camera-instellingen

Compressie van video

- meest gangbare standaard: MPEG

  Motion Picture Experts Group
- bevat verschillende standaarden voor meerdere toepassingen (HDTV ≠ videoconferentie)
- video = opeenvolging van beelden (cfr. film)
- beperkt aantal beelden (I-frame) wordt volledige gecodeerd (cf. JPEG)
- tussenliggende beelden relatief gecodeerd (afwijking)

Compressie van geluid

- best gekende formaat voor audio compressie is MP3 (ontwikkeld door MPEG)
  - misbruikt beperkingen van menselijk oor
  - verwijdert onhoorbare details, zoals o.a.
    - vlak na luide passage kan oor geen zachte passage horen
    - bepaalde frequentie verbergt stiller geluid op naburige frequentie
  - CD-kwaliteit benaderd

Inhoud

- numerieke informatie
  - gehele getallen
  - rationale getallen
  - decimale getallen
- alfanumerieke informatie (tekst)
- voorstelling en opslag van beeld en geluid
- compressie van gegevens + bestandsformaten