DIGTEC/2021-2022

Jesse op den Brouw

DIGTEC

Binaire getallen, hexadecimaal, ASCII-code, BCD-code, 7-segment display

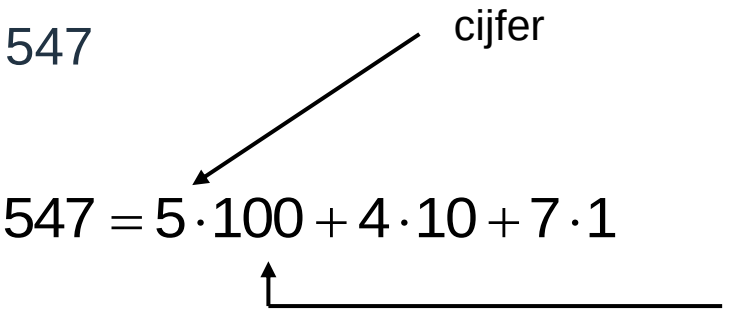
DE HAAGSE HOGESCHOOL

Decimaal talstelsel

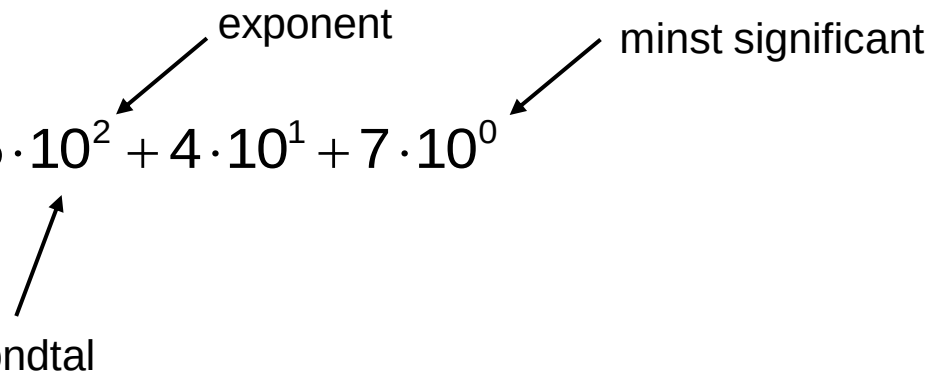
- Ons talstelsel is een zogenaamd *positioneel talstelsel*.
- Een getal bestaat uit de rij cijfers.
- Elk cijfer is één van de tien cijfers 0 t/m 9.
- Het grondtal is 10 (decimaal).
- De *positie* van het cijfer bepaalt het *gewicht*.
- Het gewicht is een macht van 10.

Decimaal talstelsel

- Neem als voorbeeld het getal 547
- Dit kan gelezen worden als: $547 = 5 \cdot 100 + 4 \cdot 10 + 7 \cdot 1$



- Of als machten van 10: $547 = 5 \cdot 10^2 + 4 \cdot 10^1 + 7 \cdot 10^0$



DE HAAGSE HOGESCHOOL

Binair talstelsel

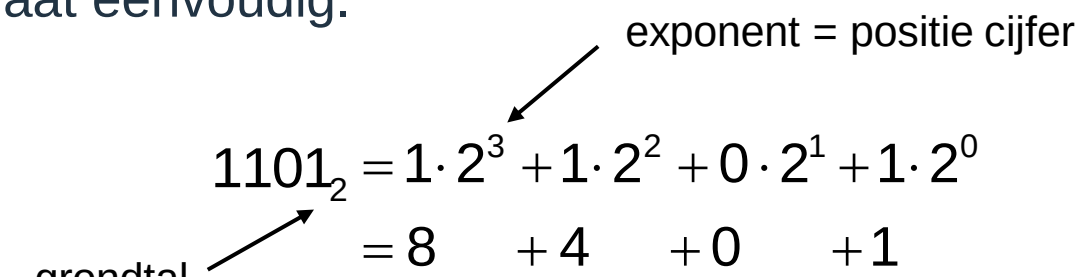
- Het binaire talstelsel bestaat slechts uit twee cijfers, 0 en 1.
- Het grondtal is 2.
- Een voorbeeld van een 8-bits getal: 11011010
- Om aan te geven dat het om een binair getal gaat, wordt een index toegevoegd:

  11011010_2

  11011010_{BIN}

Binair talstelsel

- Het omrekenen van een binair getal naar het decimale equivalent gaat eenvoudig.



$$1101_2 = 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0$$
$$= 8 + 4 + 0 + 1$$
$$= 13_{10}$$

$$11011010_2 = 1\cdot 2^7 + 1\cdot 2^6 + 1\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^1$$
$$= 128 + 64 + 16 + 8 + 2$$
$$= 218_{10}$$

DE HAAGSE HOGESCHOOL

Decimaal naar decimaal

- Hoe zet je een decimaal getal om in een decimaal getal?
- Door herhaald delen door 10:

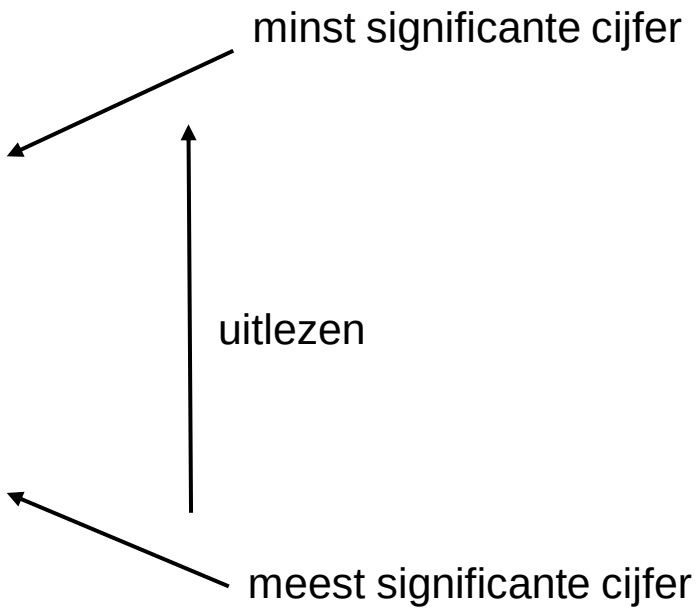
23561 ÷ 10	=	2356 rest	1
2356 ÷ 10	=	235 rest	6
235 ÷ 10	=	23 rest	5
23 ÷ 10	=	2 rest	3
2 ÷ 10	=	0 rest	2
0			

- Deze techniek toepassen voor omzetting naar binaire getallen.

Decimaal naar binair

- Als voorbeeld wordt 53_{10} omgezet:

53 ÷ 2	=	26 rest 1
26 ÷ 2	=	13 rest 0
13 ÷ 2	=	6 rest 1
6 ÷ 2	=	3 rest 0
3 ÷ 2	=	1 rest 1
1 ÷ 2	=	0 rest 1
0		



- Het binaire equivalent is dan 110101_2

DE HAAGSE HOGESCHOOL

Decimaal naar binair

- Voor het omzetten van een decimaal getal naar het binaire equivalent is herhaalde deling door 2 nodig:

1) Ga uit van het decimale getal dat omgezet moet worden.
2) Deel het gehele getal door 2.
3) Schrijf de restwaarde (0 of 1) op en ga verder met het hele deel.
4) Is het gehele deel groter dan 0? Dan naar stap 2.
5) Lees het binaire getal af uit de restwaarden.

Er is ook nog een andere methode (“passen en meten”), die wordt niet besproken.

Hexadecimaal

- Over binaire getallen kunnen wat opmerkingen gemaakt worden:
- De lengte van binaire getallen wordt snel groter (factor 3,3 t.o.v. decimale getallen),
- Grote binaire getallen zijn lastig te overzien, gevoelig voor fouten,
- Binaire getallen worden vaak geschreven in blokken van 4 bits, gescheiden door een punt. Leidende nullen worden dan ook geschreven.
- Om binaire getallen goed te kunnen overzien wordt een ander talstelsel gebruikt: hexadecimaal.

Hexadecimaal

- Het hexadecimale talstelsel is een 16-tallig stelsel.
- Er zijn dus 16 “cijfers”.
- De eerste tien cijfers zijn identiek aan die van het decimale systeem.
- De overige zes cijfers worden weergegeven door de eerste 6 letters uit het alfabet.

Hexadecimaal

- Dus:

  0 t/m 9 hebben de waarden 0 t/m 9

  A heeft de waarde 10

  B heeft de waarde 11

  C heeft de waarde 12

  D heeft de waarde 13

  E heeft de waarde 14

  F heeft de waarde 15

Hexadecimaal

- Een voorbeeld van een hexadecimaal getal:

$$3FA5_{16}$$

- Dit kan worden omgezet naar een decimaal getal:

$$3FA5_{16} = 3 \cdot 16^3 + 15 \cdot 16^2 + 10 \cdot 16^1 + 5 \cdot 16^0 = 16293_{10}$$

- Merk op: hetzelfde systeem als decimaal en binair, nu echter met grondtal 16.

Decimaal naar hexadecimaal

- Omzetten van decimaal naar hexadecimaal gaat op eenzelfde wijze als omzetten van decimaal naar binair, alleen moet er nu gedeeld worden door 16:

40796 ÷ 16	=	2549 rest 12	→	C
2549 ÷ 16	=	159 rest 5		
159 ÷ 16	=	9 rest 15	→	F
9 ÷ 16	=	0 rest 9		
0				

- Het hexadecimale equivalent is dus $9F5C_{16}$

DE HAAGSE HOGESCHOOL

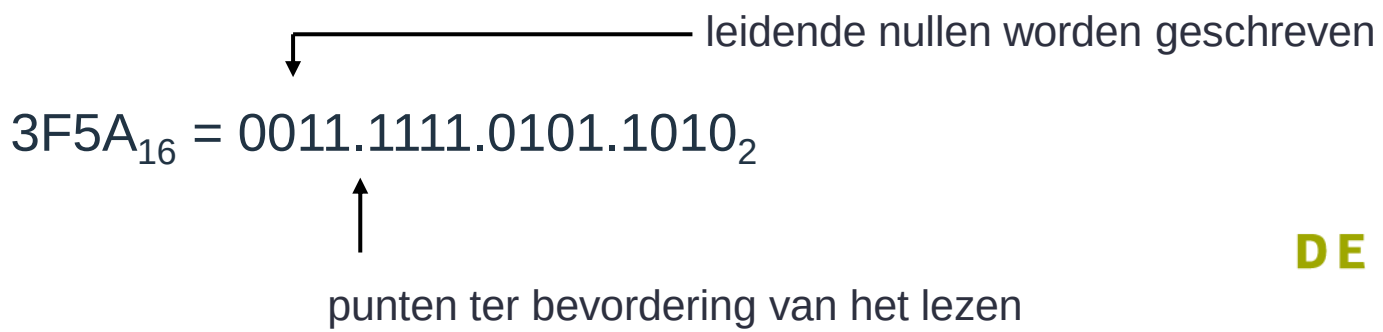
Tabel

- De eerste 17 getallen in decimaal, binair en hexadecimaal.

decimaal	binair	hexadecimaal
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

Hexadecimaal en binair

- Het hexadecimale systeem heeft als grondtal 16. Dit is een macht van 2 ($2^4 = 16$).
- Het is dus eenvoudig om een hexadecimaal getal om te zetten in een binair getal.
- Elk hexadecimaal cijfer kan worden geschreven met precies vier bits.



DE HAAGSE HOGESCHOOL

Binaire breuken

- Binaire breuken zijn eenvoudig te verwerken. Het werkt hetzelfde als met gehele getallen, alleen zijn de exponenten nu negatief.

$$0{,}1101_2 = 2^{-1} + 2^{-2} + 2^{-4} = 0{,}8125_{10}$$

- De komma geeft de scheiding aan tussen het gehele deel en de fractie.

- Binaire getallen kunnen een samenstel zijn van een geheel deel en een fractie:

$$011{,}1101_2 = 2^{1} + 2^{0} + 2^{-1} + 2^{-2} + 2^{-4} = 3{,}8125_{10}$$

Binaire breuken

- Van decimaal naar binair is lastiger, hier is herhaald vermenigvuldigen nodig. Een voorbeeld:

noteer de gehele getallen na vermenigvuldiging,

ga door met de fractie

0,8125 x 2	=	1,625	→ 1
0,625 x 2	=	1,25	→ 1
0,25 x 2	=	0,5	→ 0
0,5 x 2	=	1,0	→ 1
0			

uitlezen

stop indien 0

- Het antwoord is dan $0{,}1101_2$.

DE HAAGSE HOGESCHOOL

Binaire breuken

- Niet alle eindige decimale breuken kunnen precies worden weergegeven in het binaire systeem:

0,6 x 2	=	1,2	$\rightarrow$ 1
0,2 x 2	=	0,4	$\rightarrow$ 0
0,4 x 2	=	0,8	$\rightarrow$ 0
0,8 x 2	=	1,6	$\rightarrow$ 1
0,6 ...			

- Het antwoord is dan $0{,}100110011001....._2$.

- Dus $0{,}6_{10}$ *kan niet exact worden weergegeven in het binaire talstelsel!*

Afronden binaire breuken

- Afronden vindt plaats op basis van het eerste niet weer te geven cijfer na de komma:

  0: afronden naar beneden
  1: afronden naar boven

  $0{,}6_{10} = 0{,}1|\underline{0}... = 0{,}1_2$ (na 1 bit)
  $0{,}6_{10} = 0{,}100|\underline{1}... = 0{,}101_2$ (na 3 bits)
  $0{,}6_{10} = 0{,}10011001|\underline{1}... = 0{,}10011010_2$ (na 8 bits)

  $0{,}6_{10} \approx 0{,}10011010_2 \rightarrow 0{,}10011010_2 = 0{,}6015625_{10}$ (+0,26%)

Noot: afronden volgens de Citogroep

Benodigde aantal bits voor specifiek getal

- Om het decimale getal M weer te geven zijn tenminste *n* bits nodig zodanig dat:

$$M \leq 2^n - 1$$

- Na enig rekenwerk volgt dat:

$$n = \frac{\log(M+1)}{\log 2}$$

- Het resultaat moet worden afgerond op het laagste gehele getal dat groter is dan of gelijk is aan het oorspronkelijke getal:

$$n = \left\lceil \frac{\log(M+1)}{\log 2} \right\rceil$$

Benodigde aantal bits

- Om een decimaal getal van *m* cijfers weer te geven zijn tenminste *n* bits nodig zodanig dat:

$$10^m \leq 2^n$$

- Daaruit volgt dat:

$$n \geq \frac{m}{\log 2}$$

- Het resultaat moet worden afgerond op het laagste gehele getal dat groter is dan of gelijk is aan het oorspronkelijke getal:

$$n = \left\lceil \frac{m}{\log 2} \right\rceil$$

Bereik getallen

- In de onderstaande opsomming een aantal voorkomende bitbreedtes.

4 bits	$0 \leq g \leq 15$
8 bits	$0 \leq g \leq 255$
10 bits	$0 \leq g \leq 1.023$
12 bits	$0 \leq g \leq 4.095$
16 bits	$0 \leq g \leq 65.535$
24 bits	$0 \leq g \leq 16.777.215$
32 bits	$0 \leq g \leq 4.294.967.295$
64 bits	$0 \leq g \leq 18.446.744.073.709.551.615$
128 bits	$0 \leq g \leq 340.282.366.920.938.463.463.374.607.431.768.211.455$

BCD-code

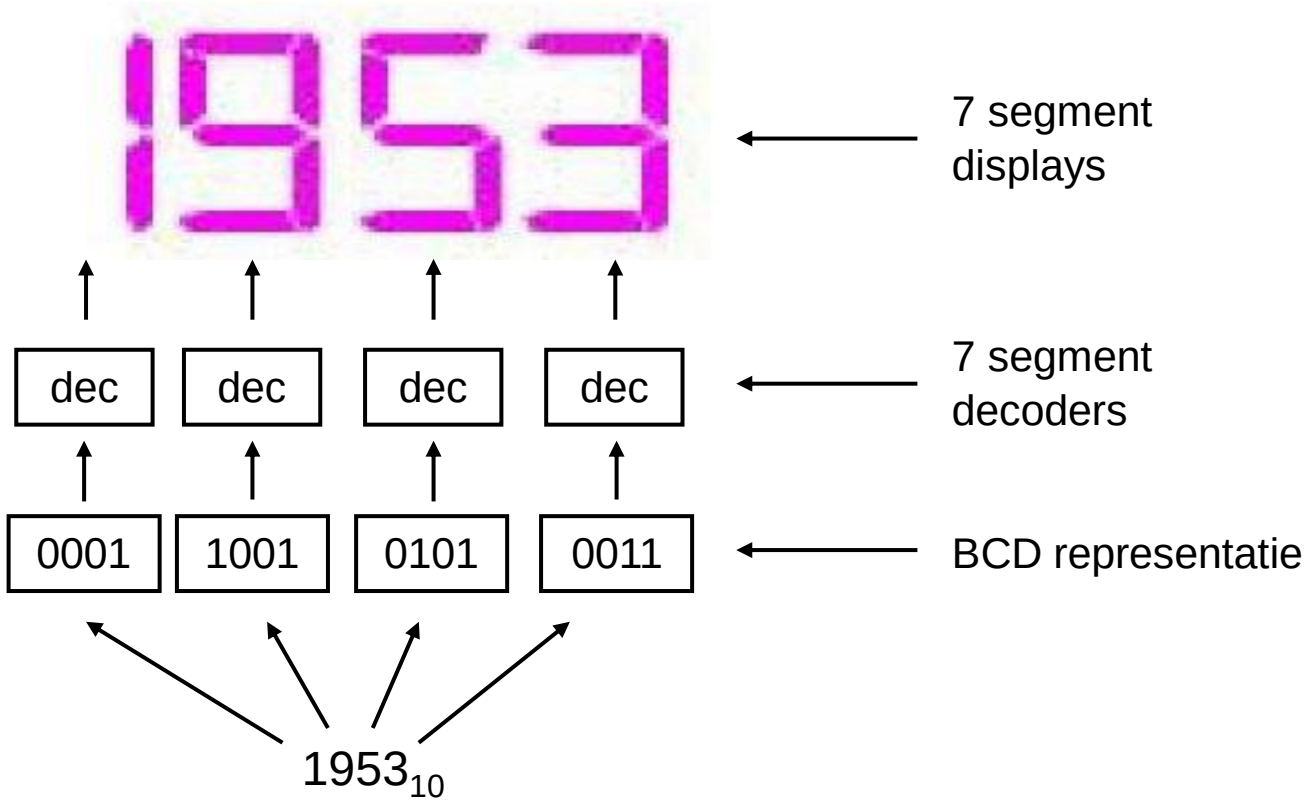
- Het nadeel van binaire getallen is dat deze niet direct kunnen worden afgebeeld als decimale getallen, immers een computer werkt met binaire getallen.
- Bij het omzetten van een binair getal naar een decimaal getal moet herhaald gedeeld worden door $10_{10} = 1010_2$ en de restwaarde stelt dan één decimaal cijfer voor.
- Een digitale schakeling voor dit probleem kost veel poorten.
- Slimmer is om een codering te gebruiken waarbij een decimaal cijfer wordt opgeslagen in vier bits.

BCD-code

- Deze codering wordt *Binary Coded Decimal* genoemd.
- Elk BCD-cijfer bestaat uit vier bits (*nibble*).
- Van de 16 combinaties worden er 10 gebruikt (0 t/m 9) en 6 niet (10 t/m 15).
- Rekenschakelingen voor BCD-gecodeerde getallen zijn lastig.
- Het afbeelden op bijvoorbeeld 7-segment displays gaat daarentegen weer heel gemakkelijk.

BCD-code

- BCD-code kan met behulp van een 7-segment decoder eenvoudig op een 7-segment display worden afgebeeld.



DE HAAGSE HOGESCHOOL

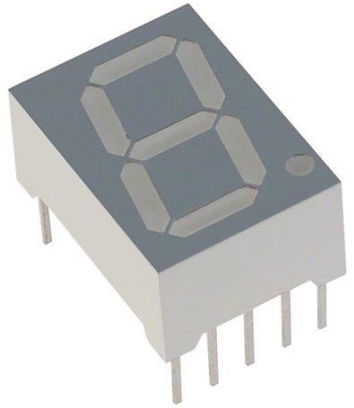
ÀSCII-code

- Naast numerieke gegevens wisselen computers ook tekst uit. Dat moet op een gestandaardiseerde manier.
- De ASCII-code (1963, 7 bits) geeft de coderingen aan voor letters, cijfers en leestekens. Zo wordt de A gecodeerd als 41_{16} (65_{10}).
- Daarnaast wordt ook een aantal *besturingstekens* gecodeerd.
  - LF = $0A_{16}$, CR = $0D_{16}$, BS = 08_{16}

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	del

7-segment display

- De 7-segment display*) is al jaren lang het middel om getallen af te beelden.
- De display bestaat uit zeven segmenten (meestal uitgevoerd als leds) en een punt (ook uitgevoerd als led).
- De zeven segmenten zijn zo gecon-strueerd dat ze samen de tien cijfers kunnen weergeven.
- Het is zelfs mogelijk een aantal letters weer te geven.



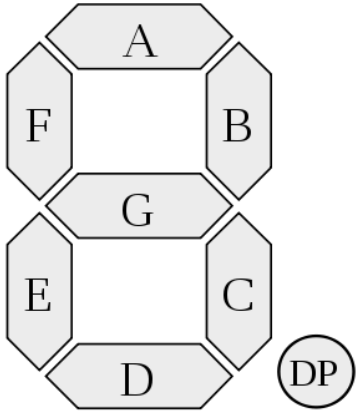
7-segment display

- Om tien verschillende cijfers weer te kunnen geven zijn vier bits nodig
- De bitcombinaties 0000 t/m 1001 worden hiervoor gebruikt.
- De bitcombinaties 1010 t/m 1111 worden niet gebruikt*).
- Dit komt precies overeen met één BCD-cijfer.

*) Voor deze combinaties kunnen ook letters worden gebruikt of worden de segmenten uitgeschakeld. Dit wordt *blanking* genoemd.

7-segment display

- Om cijfers weer te geven moeten de segmenten worden aangestuurd.
- Elk segment kan aan of uit staan, dus is een digitale schakeling te ontwerpen die dit doet.
- Dit is te realiseren met een schakeling die zeven (acht) uitgangen heeft.
- Hiervoor wordt aan de segmenten een letter toegekend, zie figuur.



DE HAAGSE HOGESCHOOL

7-segment display

- Vervolgens kunnen de tien cijfers worden geconstrueerd.

- Voor het segment A wordt dit op de volgende slides uitgewerkt.
- Aanname: led brandt bij logische 1.

7-segment decoder

- We stellen een waarheidstabel op met de ingangscombinaties 0000 t/m 1001.
- Aan elke ingangscombinatie kennen we een variabele y toe met indexnummer (y_0 t/m y_9).
- Bij elke ingangscombinatie wordt de waarde van segment A genoteerd.
- De combinaties 1010 t/m 1111 doen niet mee.

#	x_3	x_2	x_1	x_0	Y	A
0	0	0	0	0	y_0	1
1	0	0	0	1	y_1	0
2	0	0	1	0	y_2	1
3	0	0	1	1	y_3	1
4	0	1	0	0	y_4	0
5	0	1	0	1	y_5	1
6	0	1	1	0	y_6	1
7	0	1	1	1	y_7	1
8	1	0	0	0	y_8	1
9	1	0	0	1	y_9	1

7-segment decoder

- We bouwen de schakeling in twee delen op.
- Het eerste deel zorgt voor de logische schakelingen voor y_0 t/m y_9.
- Het tweede deel zorgt voor de logische schakeling voor segment A.
- Deze laatste wordt opgebouwd uit y_0 t/m y_9.

#	x_3	x_2	x_1	x_0	Y	A
0	0	0	0	0	y_0	1
1	0	0	0	1	y_1	0
2	0	0	1	0	y_2	1
3	0	0	1	1	y_3	1
4	0	1	0	0	y_4	0
5	0	1	0	1	y_5	1
6	0	1	1	0	y_6	1
7	0	1	1	1	y_7	1
8	1	0	0	0	y_8	1
9	1	0	0	1	y_9	1

7-segment decoder

- De functie voor combinatie 0000 is:

$$y_0 = \overline{x_3} \cdot \overline{x_2} \cdot \overline{x_1} \cdot \overline{x_0}$$

- En voor 0001:

$$y_1 = \overline{x_3} \cdot \overline{x_2} \cdot \overline{x_1} \cdot x_0$$

- Tot en met 1001:

$$y_9 = x_3 \cdot \overline{x_2} \cdot \overline{x_1} \cdot x_0$$

#	x_3	x_2	x_1	x_0	Y	A
0	0	0	0	0	y_0	1
1	0	0	0	1	y_1	0
2	0	0	1	0	y_2	1
3	0	0	1	1	y_3	1
4	0	1	0	0	y_4	0
5	0	1	0	1	y_5	1
6	0	1	1	0	y_6	1
7	0	1	1	1	y_7	1
8	1	0	0	0	y_8	1
9	1	0	0	1	y_9	1

7-segment decoder

- Nu alle functies voor y_0 t/m y_9 bekend zijn, kan de functie voor segment A uitgewerkt worden.
- Segment A moet gaan branden voor elke ingangscombinatie waarvoor bij A in de waarheidstabel een logische 1 staat.
- Dat is dus bij y_0, y_2, y_3, y_5, y_6, y_7, y_8 en y_9. De functie wordt dan:

$$A = y_0 + y_2 + y_3 + y_5 + y_6 + y_7 + y_8 + y_9$$

#	x_3	x_2	x_1	x_0	Y	A
0	0	0	0	0	y_0	1
1	0	0	0	1	y_1	0
2	0	0	1	0	y_2	1
3	0	0	1	1	y_3	1
4	0	1	0	0	y_4	0
5	0	1	0	1	y_5	1
6	0	1	1	0	y_6	1
7	0	1	1	1	y_7	1
8	1	0	0	0	y_8	1
9	1	0	0	1	y_9	1

7-segment decoder

- Dit levert wel veel logica op:

  4x NOT, 8x AND4, 1x OR8

- Slimmer is om naar de 0-en te kijken. De schakeling is 0 bij y_1 en y_4. Dus:

$$A(0) = y_1 + y_4$$

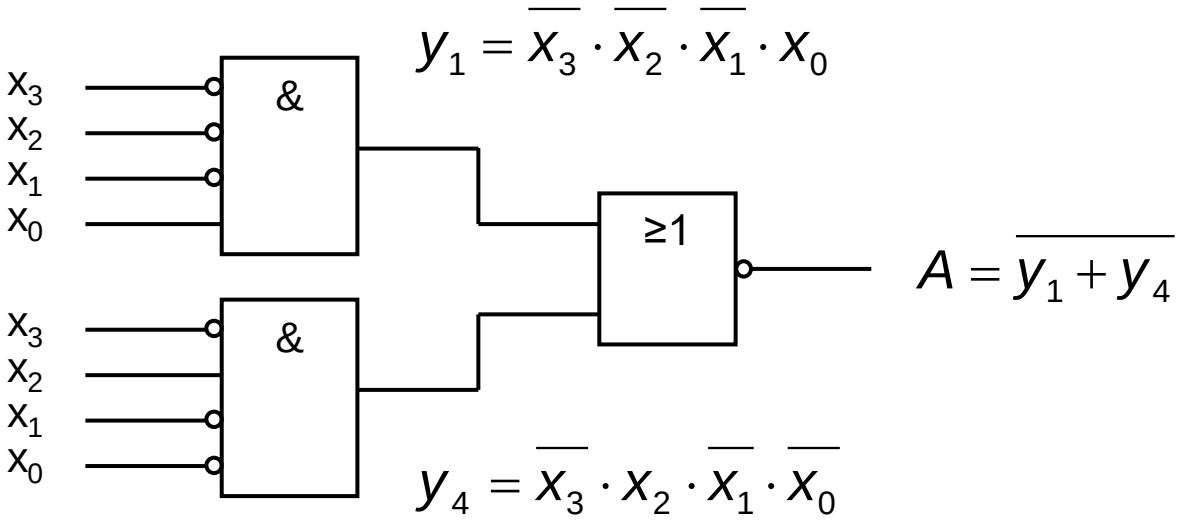
- Om 1-en te krijgen met de functie geinverteerd worden:

$$A = \overline{A(0)} = \overline{y_1 + y_4}$$

#	x_3	x_2	x_1	x_0	Y	A
0	0	0	0	0	y_0	1
1	0	0	0	1	y_1	0
2	0	0	1	0	y_2	1
3	0	0	1	1	y_3	1
4	0	1	0	0	y_4	0
5	0	1	0	1	y_5	1
6	0	1	1	0	y_6	1
7	0	1	1	1	y_7	1
8	1	0	0	0	y_8	1
9	1	0	0	1	y_9	1

7-segment decoder

- Het schema wordt dan:



- NOT-poorten als geinverteerde ingangen.

DE HAAGSE HOGESCHOOL

Opgaven

- Zet het getal 543 om naar binair.
- Hoeveel bit zijn er minimaal nodig om het getal 647384 binair te representeren

let’s change