Universiteti i Prishtinës "Hasan Prishtina" Kosovë

Arkitektura e kompjuterit

Pjesa 2 – Paraqitja e të dhënave në sistemet kompjuterike

Prof. Asoc. Dr. Ermir Rogova

Objektivat

- T'i njohim kodet më karakteristike të simboleve.
- Të kuptojmë se si të dhënat ruhen në memorien e kompjuterit, si transmetohen nëpër kanale komunikimi dhe si ruhen në disqe.
- Ta qartësojmë kuptimin e kodeve për detektimin e gabimeve dhe kodeve për përmirësimin e gabimeve.

Kodet e simboleve

- Njehsimet nuk janë të dobishme përderisa rezultatet e tyre nuk mund të prezantohen në një formë kuptimplote për njerëzit.
- Ne duhet t'i ruajmë rezultatet e njehsimeve dhe të ofrojmë mundësi për pranimin e të dhënave hyrëse.
- Pra, simbolet e kuptueshme për njeriun duhet të konvertohen në vargje të bitëve të kuptueshme për kompjuterin duke përdorur një lloj skeme për kodimin e simboleve.

BCD

- Me zhvillimin e kompjuterëve janë zhvilluar edhe kodet e simboleve
- Kapaciteti i shtuar i memories kompjuterike dhe pajisjeve për ruajtjen e të dhënave bën të mundur shfrytëzimin e kodeve më të pasura të simboleve
- Sistemet e para të kodimit të simboleve kanë përdorur 6 bit
- Binary-coded decimal (BCD) ishte njëri nga këto kode. U përdor nga IBM në vitet 1950 dhe 1960.

EBCDIC

- Më 1964, BCD u zgjerua nga 6 në kodin 8 bitësh Extended Binary-Coded Decimal Interchange Code (EBCDIC)
- EBCDIC ishte në mesin e kodeve të para që përkrahin shkronjat e mëdha dhe të vogla, si dhe simbolet speciale dhe kontrolluese
- EBCDIC dhe BCD ende përdoren në kompjuterët që i prodhon IBM

ASCII

- Prodhuesit tjerë të kompjuterëve zgjodhën kodin 7-bitësh ASCII (American Standard Code for Information Interchange) si zëvendësim për kodet 6-bitëshe
- Derisa BCD dhe EBCDIC burojnë nga kodet që përdoreshin për kartela të perforuara, ASCII buronte nga kodet që përdoreshin në telekomunikacion (Telex)
- Deri vonë, ASCII ishte kodi dominant jashtë IBM

Unicode (1/2)

- Shumë sisteme bashkëkohore e shfrytëzojnë Unicode-in, një sistem 16-bitësh që mund t'i kodojë simbolet e çdo gjuhe botërore
- Gjuha programore Java dhe disa sisteme operative e përdorin Unicode-in si kod të zakonshëm të simboleve (default)
- Hapësira e Unicode ndahet në gjashtë pjesë. Pjesa e parë është për kodet e alfabeteve perëndimore, ku përfshihen gjuhë me alfabete latine, cirilike, si dhe greqishtja

Unicode (2/2)

- Simbolet e Unicode me numra më të vegjël rendorë përfshijnë kodin ASCII.
- Simbolet me numra të mëdhenj rendorë shërbejnë për kode që i definon shfrytëzuesi.

Character Types	Language	Number of Characters	Hexadecimal Values
Alphabets	Latin, Greek, Cyrillic, etc.	8192	0000 to 1FFF
Symbols	Dingbats, Mathematical, etc.	4096	2000 to 2FFF
CJK	Chinese, Japanese, and Korean phonetic symbols and punctuation.	4096	3000 to 3FFF
Han	Unified Chinese, Japanese, and Korean	40,960	4000 to DFFF
	Han Expansion	4096	E000 to EFFF
User Defined		4095	F000 to FFFE

Tabela ASCII

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

Kodet për incizimin dhe transmetimin e të dhënave (1/2)

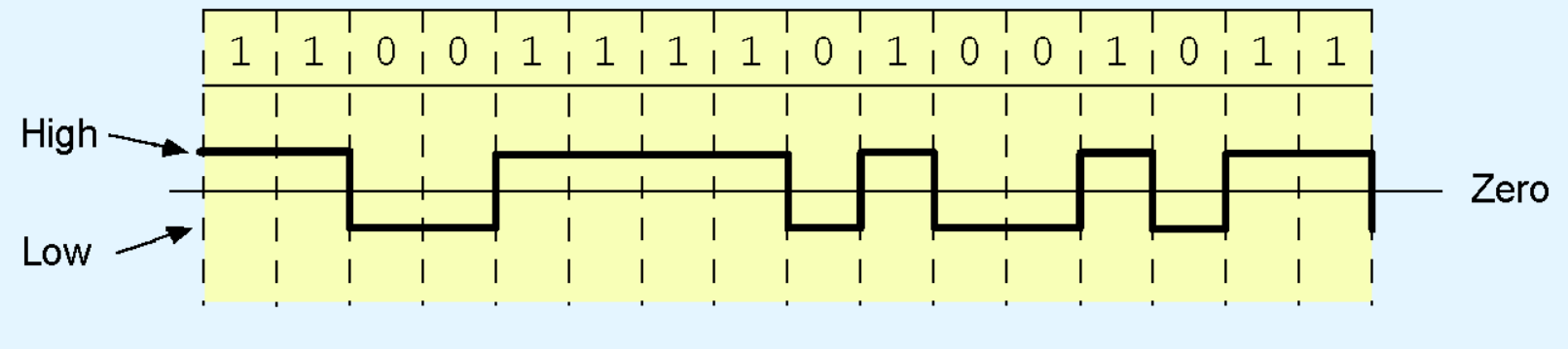
- Kur kodet e simboleve ose vlerat numerike ruhen në memorien e kompjuterit, vlerat e tyre janë të qarta
- Kjo nuk vlen gjithnjë për rastet kur të dhënat ruhen në njësi të memories periferike ose transmetohen në distancë më të madhe se 50 cm
- Për shkak të parregullësive fizike të mediumit për ruajtje ose transmetim, bajtët mund të ngatërrohen
- Gabimet reduktohen me përdorimin e metodave të përshtatshme të kodimit si dhe me përdorimin e teknikave të ndryshme për detektimin e gabimeve

Kodet për incizimin dhe transmetimin e të dhënave (2/2)

- Transmetimi i të dhënave nëpër medium bëhet me anë të impulseve të rrymës të voltazhit "të lartë" ose "të ulët"
- Ruajtja e të dhënave në mediumin magnetik bëhet përmes ndryshimit të polaritetit magnetik të mediumit
- Këto ndryshime të polaritetit quhen *kthime të fluksit*
- Periudha kohore gjatë të cilës transmetohet biti ose zona e mediumit magnetik ku ruhet biti quhet *qelizë binare*

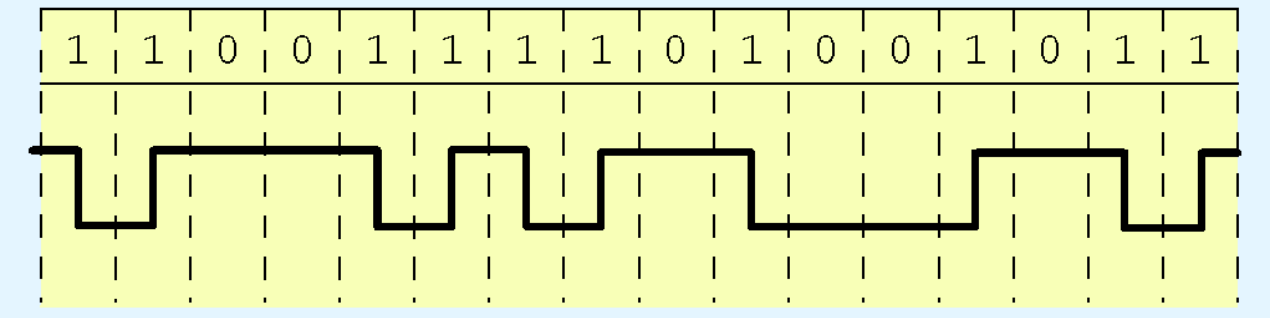
NRZ

- Kodi më i thjeshtë për incizim dhe transmetim është "non-return-to-zero (NRZ) code".
- NRZ e kodon 1 si "të lartë", ndërsa 0 si "të ulët"
- Më poshtë është dhënë kodimi i fjalës ok në ASCII.



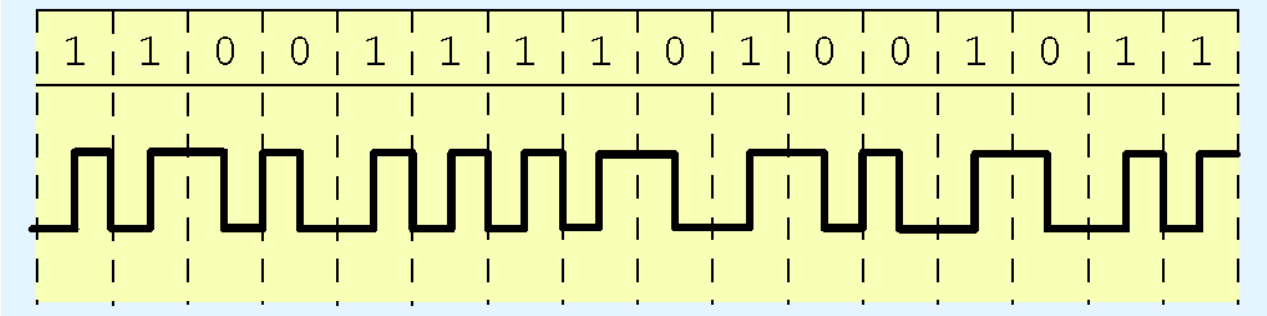
NRZI

- Problem i NRZ qëndron në faktin se vargjet e gjata të zerove dhe njësheve shkaktojnë humbje sinkronizimi
- Non-return-to-zero-invert (NRZI) zvogëlon humbjen e sinkronizimit duke ofruar një kalim (qoftë nga "ulët"-"lartë", apo nga "lartë"-"ulët") për çdo 1 binar



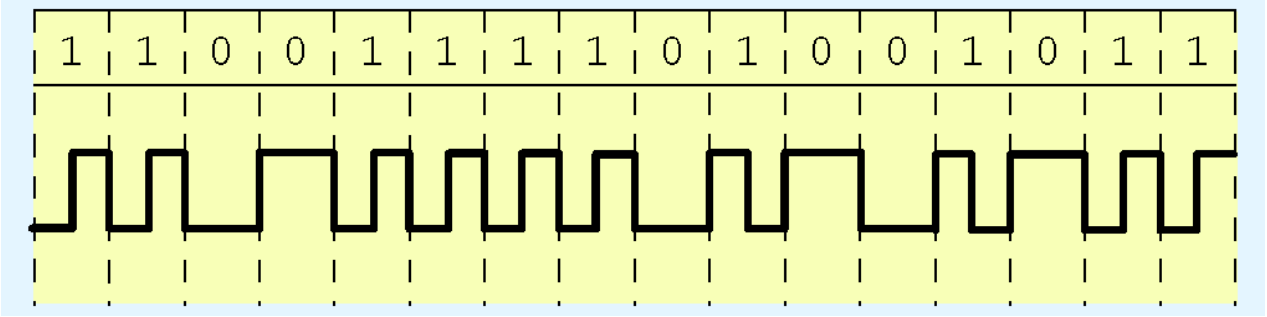
Manchester

- Edhepse e parapengon humbjen e sinkronizimit tek vargjet e gjata të njësheve, NRZI nuk bën asgjë për ta penguar humbjen e sinkronizimit tek vargjet e gjata të zerove
- Kodimi "Manchester" (i njohur edhe si modulimi fazor) e pengon këtë problem duke e koduar njëshin me kalim (transfer) "lartë", ndërsa zeron me kalim "poshtë"



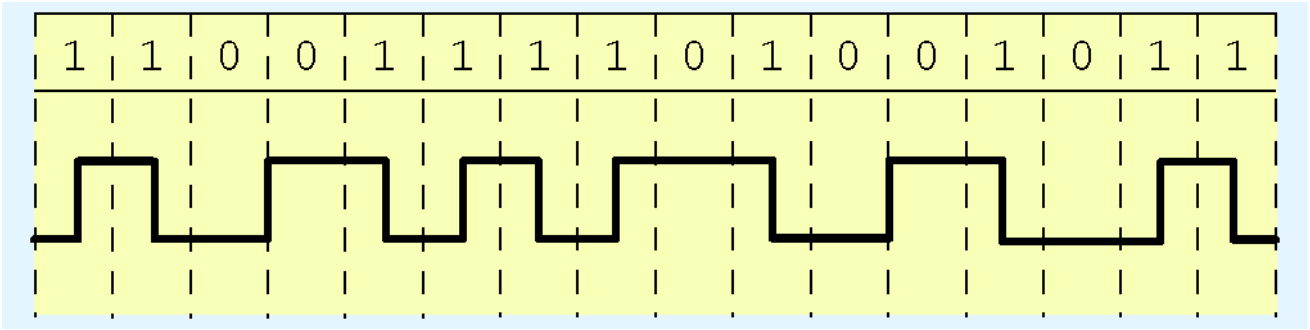
Frequency modulation (FM)

- Për shumë vite, kodi "Manchester" dominonte në rrjetat lokale
- Sidoqoftë, ky kod shkakton tepricë komunikimi pasi ka kalime për çdo qelizë binare
- Një metodë më efektive e kodimit është "frequency modulation (FM) code". Tek FM, kalimi bëhet në kufirin e çdo qelize binare. Ndërkaq, qelizat me 1 binar e bëjnë kalimin në mes të qelizës



MFM

- Në shikim të parë FM është më i papërshtatshëm se "Manchester", sepse kërkon kalim në çdo kufi të qelizës binare
- Sikur t'i eliminojmë disa nga këto kalime do të kishim një kod shumë më ekonomik
- FM i modifikuar (MFM) e bën pikërisht këtë. Kalimi në kufi të qelizës bëhet vetëm nëse qelizat fqinje përmbajnë zero
- Qeliza MFM me 1 binar ka kalim në mes, si në rastin e FM



RLL

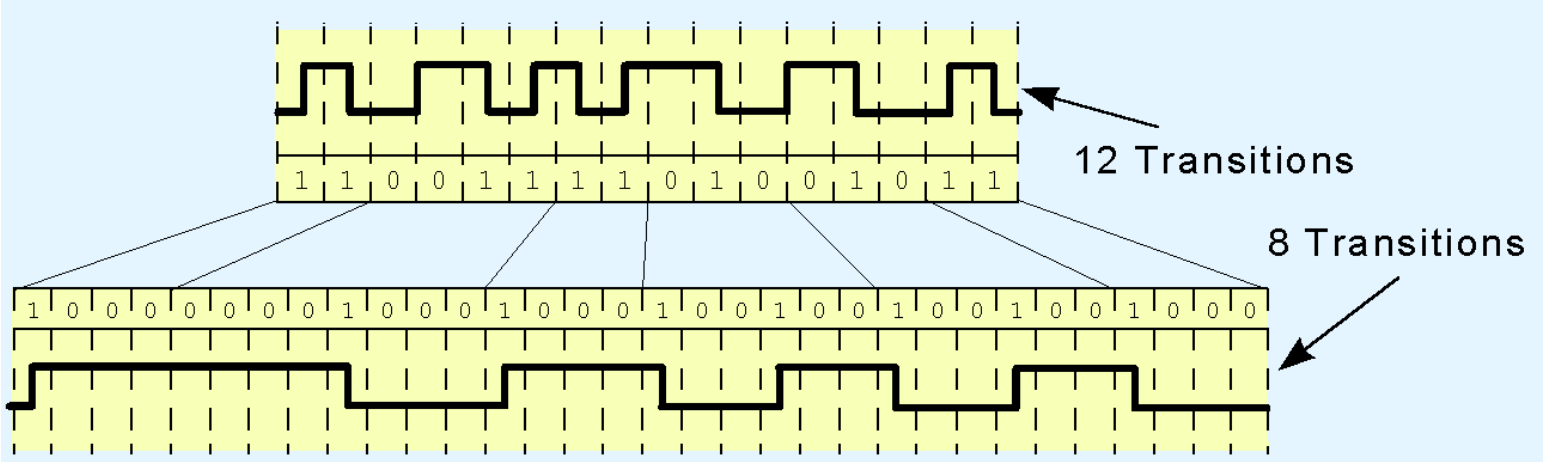
- Sfida kryesore për incizimin dhe transmetimin e të dhënave është si të ruhet sinkronizimi pa shfrytëzuar më shumë resurse se që është e nevojshme
- Run-length-limited (RLL) është kod i disenjuar për ta reduktuar numrin e zerove dhe njësheve të njëpasnjëshme
  - Disa bite më shumë shtohen në kod
  - Por edhe me këto bite shtesë RLL është shumë efektiv

RLL(d,k)

- Kodi RLL(d,k) përcakton minimumin prej d dhe maksimumin prej k zerove të njëpasnjëshme ndërmjet çdo çifti të njësheve të njëpasnjëshme
- RLL(2,7) ka qenë kodi më i përdorur për ruajtjen e të dhënave në disqe për shumë vite
- Një simbol i koduar në RLL(2,7) përmban më shumë qeliza binare se sa simboli korrespondues në ASCII ose EBCDIC
- Sidoqoftë, metoda e kodimit i lejon qelizat të jenë më të vogla, pra më afër njëra-tjetrës se në rastin e MFM ose ndonjë kodi tjetër

RLL(2,7)

- Kodi RLL(2,7) për OK është dhënë më poshtë dhe është krahasuar me MFM. Kodi RLL (poshtë) përmban 25% më pak kalime se sa kodi MFM (lartë)



Detektimi dhe përmirësimi i gabimeve

- Është e pamundur që një medium për ruajtjen ose transmetimin e të dhënave të jetë 100% i sigurtë tërë kohën
- Sa më shumë bitë që vendosen në një cm^2 të hapësirës së diskut dhe sa më shumë që rritet shpejtësia e komunikimit, aq më shumë rritet gjasa për gabime
- Prandaj, detektimi i gabimeve dhe përmirësimi i tyre është kritik për transmetimin, ruajtjen dhe leximin korrekt të të dhënave

Shifrat për verifikim

- Shifrat për verifikim në fund të një numri të gjatë ofrojnë një lloj mbrojtje nga gabimet në të dhëna
  - Numrat e fundit në barkode UPC, ISBN, CC, ID, janë shifra për verifikim
- Ndërkaq, vargjet më të gjata të të dhënave kërkojnë mekanizma më ekonomikë dhe më të sofistikuar detektimi
- Kodet CRC (Cyclic redundancy checking) ofrojnë mundësi për detektimin e gabimeve në blloqe më të mëdha të të dhënave

Shifrat për verifikim – ISBN-10

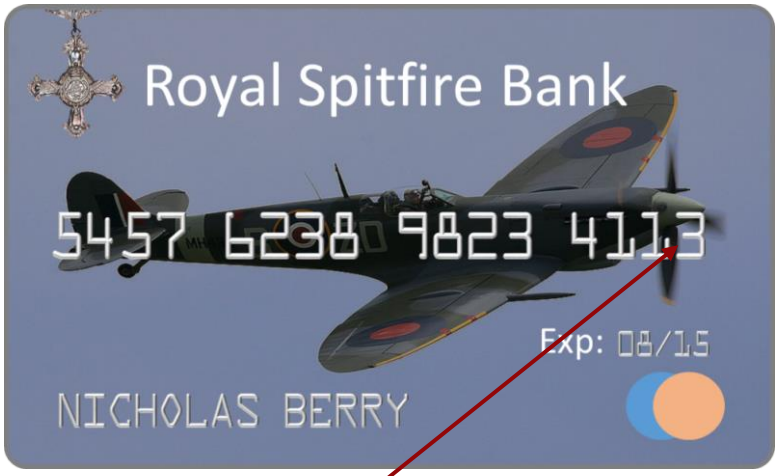
- ISBN (International Standard Book Number) është numër unik që i jepet një libri.
- ISBN-10:
  - Numri ka 9 shifra informative dhe përfundon me një shifër verifikuese
  - Supozojmë që shifrat janë "abcdefghi-j" ku j është verifikuesi. Atëherë ai llogaritet me formulën:
    j = ([a b c d e f g h i] * [1 2 3 4 5 6 7 8 9]) mod 11
- (3*1)+(8*2)+(8*3)+(0*4)+(5*5)+(3*6)+(1*7)+(0*8)+(1*9)
- 3+16+24+0+25+18+7+0+9 = 102
- X= 102 mod 11 = 3

Shifrat për verifikim – ISBN-13

- ISBN (International Standard Book Number) është numër unik që i jepet një libri.
- ISBN-13:
  - Numri ka 12 shifra informative dhe përfundon me një shifër verifikese
  - Supozojmë që shifrat janë "abcdefghijkl-m" ku m është verifikuesi. Atëherë ai llogaritet me formulën:
    m = 10 - ([a b c d e f g h i j k l] * [1 3 1 3 1 3 1 3 1 3 1 3]) mod 10
  - (9*1)+(7*3)+(8*1)+(1*3)+(9*1)+(1*3)+(1*1)+(2*3)+(2*1)+(3*3)+(1*1)+(3*3)
- 9+21+8+3+9+3+1+6+2+9+1+9= 81
- X=10 –(81 mod 10) = 9

Shifrat për verifikim - CC

- Algoritmi i Luhn-it



2x		2x		2x		2x		2x		2x		2x		2x		
5	4	5	7	6	2	3	8	9	8	2	3	4	1	1	X	
2x5=10		2x5=10		2x6=12		2x3=6		2x9=18		2x2=4		2x4=8		2x1=2		
(1+0)	1x4=4	1	1x7=7	(1+2)	1x2=2	6	1x8=8	(1+8)	1x8=8	4	1x3=3	8	1x1=1	2		=34
	4		7		2		8		8		3		1			=33
																=67

- X= 10 – (67 Mod 10) = 3

Detektimi dhe përmirësimi i gabimeve - CRC

- Shumat verifikuese (checksums) dhe CRC janë shembuj të detektimit sistematik të gabimit
- Te detektimi sistematik i gabimit një grup i biteve kontrolluese shtohet në fund të çdo blloku të transmetuar të të dhënave
  - Ky grup i bitëve quhet sindrom
- CRC-të janë polinome mbi fushën e kongruencave sipas modulit 2

Teoria matematikore e polinomeve sipas modulit 2 nuk është objekt shqyrtimi i këtij kursi, prandaj do t'i shfrytëzojmë rezultatet e gatshme.

Detektimi dhe përmirësimi i gabimeve

- Gabimet në transmetimin e të dhënave mund të përmirësohen pasi të jenë detektuar
  - Kërkohet nga dërguesi ta transmetojë mesazhin edhe një herë
- Në memorien e kompjuterit dhe memorien periferike kjo nuk është e mundur
  - Shumë shpesh kopja e vetme e të dhënave është pikërisht ajo që ndodhet në memorie ose në disk
- Prandaj, për të siguruar integritetin e të dhënave për një kohë më të gjatë, përdoren kodet për përmirësimin e gabimeve (error correcting codes)

Detektimi dhe përmirësimi i gabimeve

- Kodet e Hammingut dhe Reed-Solomonit janë dy prej tyre
- Kodet e Reed-Solomonit janë të dobishme për përmirësimin e gabimeve të mëdha, të cilat paraqiten kur dëmtohen vargje të tëra të biteve fqinje
  - P.sh. CD-ROM mund të dëmtohet lehtë, prandaj ato përdorin një lloj të kodit të Reed-Solomonit
- Këtu do të flitet më hollësisht për kodet e Hammingut pasi matematika e tyre është më e thjeshtë

Detektimi dhe përmirësimi i gabimeve

- Kodi i Hammingut përbëhet prej fjalëve që fitohen duke shtuar bite pariteti një fjale që përmban të dhëna
- Distanca e Hammingut në mes të dy fjalëve të kodit paraqet numrin e bitëve për të cilin ato dy fjalë dallohen

Ky çift e ka distancën e Hammingut 3:

```
1 0 0 0 1 0 0 1
1 0 1 1 0 0 0 1
```

- Distanca minimale e Hammingut është distanca më e vogël në mes të çdo dy fjalëve të kodit

Detektimi dhe përmirësimi i gabimeve

- Distanca minimale e Hammingut Dmin përcakton kapacitetin për detektimin dhe përmirësimin e gabimit
- Që një fjalë e kodit, X, të intepretohet gabimisht si një fjalë tjetër e kodit, Y, në X duhet të bëhen të paktën Dmin gabime në bite
- Pra, për të detektuar k gabime në bite, kodi duhet ta ketë distancën e Hammingut prej të paktën k + 1

Detektimi dhe përmirësimi i gabimeve

- Kodet e Hammingut mund t'i detektojnë D_{min} - 1 gabime dhe t'i përmirësojnë $\frac{D_{min}-1}{2}$ gabime
- Pra, distanca e Hammingut prej 2k + 1 kërkohet për t'i përmirësuar k gabime në çdo fjalë me të dhëna
- Distanca e Hammingut fitohet duke shtuar një numër të përshtatshëm të biteve të paritetit një fjale me të dhëna

Detektimi dhe përmirësimi i gabimeve

- Supozojmë se e kemi një bashkësi të fjalëve n – bitëshe të kodit , me m bite të të dhënave dhe r bite pariteti
- Gabimi mund të paraqitet në cilindo nga n bitët, kështu që çdo fjalë e kodit mund të shoqërohet me n fjalë të gabuara me distancë të Hammingut 1
- Pra, për çdo fjalë kodi i kemi n + 1 vargje të bitëve: një fjalë valide dhe n të gabuara.

Detektimi dhe përmirësimi i gabimeve

- Nëse m = 8 atëherë:

  $(8 + r + 1) \leq 2^r$

  pra r >= 4.

- Pra për të ndërtuar kod me fjalë 8-bitëshe, i cili mund të përmirësojë gabimin në një bit të vetëm, duhet krijuar fjalë kodi të gjatësisë 12
- Si i japim vlera biteve të paritetit?

Detektimi dhe përmirësimi i gabimeve

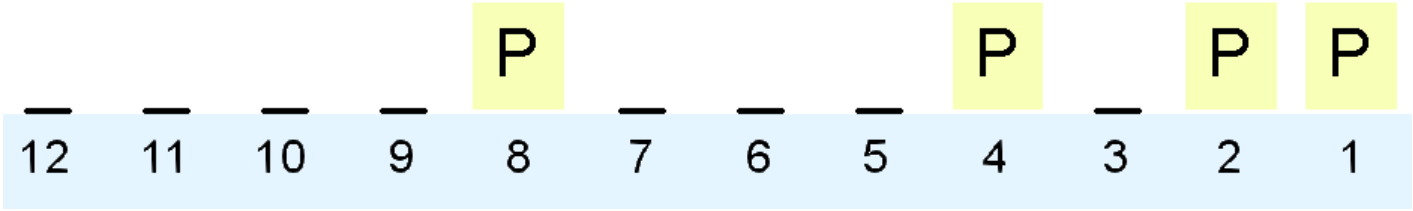
- Vërejmë se:

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

  - 1 (= 2^0) merr pjesë tek numrat tek.
  - 2 (= 2^1) merr pjesë tek numrat 2, 3, 6, 7, 10 dhe 11.
  - 4 (= 2^2) merr pjesë tek numrat 4, 5, 6, 7 dhe 12.
  - 8 (= 2^3) merr pjesë tek numrat 8, 9, 10, 11 dhe 12.

Detektimi dhe përmirësimi i gabimeve

- I numërojmë bitët prej 1-12 nga e majta.
- Vendosim nga një bit pariteti në pozitat që janë fuqi e plotë e numrit 2 (1,2,4 dhe 8)



Detektimi dhe përmirësimi i gabimeve

- Numri 2 (= 2^1) merr pjesë në numrat 2, 3, 6, 7, 10 dhe 11. Pozicioni 2 përmban bitin e paritetit për pozicionet 3, 6, 7, 10 dhe 11.
- Nëse e përdorim paritetin çift, ky është shuma e vlerave të bitëve sipas modulit 2

1	1	0	1		0	1	1		0	0	
12	11	10	9	8	7	6	5	4	3	2	1

Cilat janë vlerat për bitet tjera të paritetit?

Detektimi dhe përmirësimi i gabimeve

1	1	0	1	1	0	1	1	1	0	0	1
12	11	10	9	8	7	6	5	4	3	2	1

- Ja fjala e kompletuar e kodit
  - Biti 1 verifikon shifrat 3, 5, 7, 9,11
  - Biti 2 verifikon shifrat 2,3,6,7,10,11
  - Biti 4 verifikon shifrat 5, 6, 7,12
  - Biti 8 verifikon shifrat 9, 10, 11, 12
- Me algoritmin e Hammingut mund të detektojmë vetëm një gabim në bite, por edhe ta përmirësojmë

Detektimi dhe përmirësimi i gabimeve

1	1	0	1	1	0	1	0	1	0	0	1
12	11	10	9	8	7	6	5	4	3	2	1

- Supozojmë se gabimi paraqitet në bitin 5
- Vlerat e paritetit janë:
  - Biti 1 verifikon shifrat 3, 5, 7, 9,11 (Duhet të jetë: 0)
  - Biti 2 verifikon shifrat 2,3,6,7,10,11 (Duhet të jetë: 0)
  - Biti 4 verifikon shifrat 5, 6, 7,12 (Duhet të jetë: 0)
  - Biti 8 verifikon shifrat 9, 10, 11, 12 (Duhet të jetë: 1)

Detektimi dhe përmirësimi i gabimeve

1	1	0	1	1	0	1	0	1	0	0	1
12	11	10	9	8	7	6	5	4	3	2	1

- Kemi bite të gabuara në pozitat 1 dhe 4
- Me dy gabime në bite të paritetit e dimë se gabimi është në të dhëna e jo në bite të paritetit
- Cili bit ka gabim? Këtë e kuptojmë duke i mbledhur pozicionet e biteve të gabueshme, pra: 1 + 4 = 5
- Prandaj, nëse e ndërrojmë vlerën e bitit 5 nga 0 në 1, të gjitha bitet e paritetit do të rregullohen

Detektimi dhe përmirësimi i gabimeve

- Duke përdorur kodin e Hammingur paraqitni vargun për transmetimin e numrit 113

Detektimi dhe përmirësimi i gabimeve

- Duke përdorur kodin e Hammingur paraqitni vargun për transmetimin e numrit 113
- 113 --> 1110001
- 111x000x1xx
- 11110000111

Detektimi dhe përmirësimi i gabimeve

- Janë pranuar dy transmetime 11 bitëshe:
- 10000101111 dhe 10100010010
- Detektoni dhe përmirësoni gabimet (nëse ka) dhe gjeni çka përfaqsojnë këto kode në ASCII

Detektimi dhe përmirësimi i gabimeve

- Janë pranuar dy transmetime 11 bitëshe:
- 10000101111 dhe 10100010010
- Detektoni dhe përmirësoni gabimet (nëse ka) dhe gjeni çka përfaqsojnë këto kode në ASCII
- 10100101111 --> 1010101 --> U
- 10100000010 --> 1010000 --> P

Detektimi dhe përmirësimi i gabimeve

- Është pranuar një transmetim 15 bitësh:
- 110111000101011
- Detektoni dhe përmirësoni gabimet (nëse ka) dhe gjeni çka përfaqsojnë këto kode në Decimal

Detektimi dhe përmirësimi i gabimeve

- Është pranuar një transmetim 15 bitësh:
- 110111000101011
- Detektoni dhe përmirësoni gabimet (nëse ka) dhe gjeni çka përfaqsojnë këto kode në Decimal
- 111111000101011 --> 11111100100 --> 2020

Detektimi dhe përmirësimi i gabimeve

- Është pranuar një transmetim 30 bitësh:
- 11010110011110110000101100011
- Detektoni dhe përmirësoni gabimet (nëse ka) dhe gjeni çka përfaqsojnë këto kode në Decimal

Detektimi dhe përmirësimi i gabimeve

- Është pranuar një transmetim 30 bitësh:
- 11010110011110110000101100011
- Detektoni dhe përmirësoni gabimet (nëse ka) dhe gjeni çka përfaqsojnë këto kode në Decimal.
- 11010110011110110000101100011
- Zgjidhja: 28111912

Përmbledhje

- Kodet e simboleve janë ASCII, EBCDIC dhe Unicode
- Kodet për transmetim dhe ruajtje të të dhënave janë konstruktuar të jenë ekonomike dhe të besueshme
- Kodet për detektim dhe përmirësim të gabimeve janë të domosdoshme, sepse nuk presim që mediumet për transmetim dhe ruajtje të jenë të pagabueshme
- Kodet më të rëndësishme për detektimin dhe përmirësimin e gabimit janë: CRC, Reed-Soloman dhe Hamming

Pyetje???