CodeBash

codebash.co.uk

FREE SAMPLE · SHARE FREELY

Number Bases

Worked Solutions

Binary, denary and hexadecimal conversions.

GCSE & A-Level Computer Science

This booklet is yours to keep, print and share.

Find more revision worksheets, an online code editor, lockdown assessments and full theory revision at **codebash.co.uk**.

GCSE CS

CodeBash

Number Bases

codebash.co.uk

200 practice questions - binary, denary, hex, two's complement, bit shifts

SOLUTIONS

GCSE CS

CodeBash

Number Bases

codebash.co.uk

SOLUTIONS

SOLUTIONS

Section A: Binary to Denary (30 marks)

Convert each 8-bit binary number to its denary (decimal) equivalent.

Method - Binary to Denary:
Use column weights: 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1
Multiply each bit by its column weight, then sum the results.
Example: 10110010 = 128+32+16+2 = 178

Q	Binary	Denary	Working
1	00010010	18	*128|64|32|16|8|4|2|1 -> 16 + 2 = 18*
2	10110111	183	*128|64|32|16|8|4|2|1 -> 128 + 32 + 16 + 4 + 2 + 1 = 183*
3	01010100	84	*128|64|32|16|8|4|2|1 -> 64 + 16 + 4 = 84*
4	01101100	108	*128|64|32|16|8|4|2|1 -> 64 + 32 + 8 + 4 = 108*
5	00000001	1	*128|64|32|16|8|4|2|1 -> 1 = 1*
6	11001000	200	*128|64|32|16|8|4|2|1 -> 128 + 64 + 8 = 200*
7	01001001	73	*128|64|32|16|8|4|2|1 -> 64 + 8 + 1 = 73*
8	01010100	84	*128|64|32|16|8|4|2|1 -> 64 + 16 + 4 = 84*
9	11000001	193	*128|64|32|16|8|4|2|1 -> 128 + 64 + 1 = 193*
10	10110100	180	*128|64|32|16|8|4|2|1 -> 128 + 32 + 16 + 4 = 180*
11	10110010	178	*128|64|32|16|8|4|2|1 -> 128 + 32 + 16 + 2 = 178*
12	10101111	175	*128|64|32|16|8|4|2|1 -> 128 + 32 + 8 + 4 + 2 + 1 = 175*
13	01111001	121	*128|64|32|16|8|4|2|1 -> 64 + 32 + 16 + 8 + 1 = 121*
14	10011111	159	*128|64|32|16|8|4|2|1 -> 128 + 16 + 8 + 4 + 2 + 1 = 159*
15	11000001	193	*128|64|32|16|8|4|2|1 -> 128 + 64 + 1 = 193*
16	01000110	70	*128|64|32|16|8|4|2|1 -> 64 + 4 + 2 = 70*
17	01110100	116	*128|64|32|16|8|4|2|1 -> 64 + 32 + 16 + 4 = 116*
18	01111100	124	*128|64|32|16|8|4|2|1 -> 64 + 32 + 16 + 8 + 4 = 124*
19	11011110	222	*128|64|32|16|8|4|2|1 -> 128 + 64 + 16 + 8 + 4 + 2 = 222*
20	10010000	144	*128|64|32|16|8|4|2|1 -> 128 + 16 = 144*
21	10011101	157	*128|64|32|16|8|4|2|1 -> 128 + 16 + 8 + 4 + 1 = 157*
22	01111010	122	*128|64|32|16|8|4|2|1 -> 64 + 32 + 16 + 8 + 2 = 122*
23	00100100	36	*128|64|32|16|8|4|2|1 -> 32 + 4 = 36*
24	11101101	237	*128|64|32|16|8|4|2|1 -> 128 + 64 + 32 + 8 + 4 + 1 = 237*
25	10010010	146	*128|64|32|16|8|4|2|1 -> 128 + 16 + 2 = 146*

26	11111101	**253**	*128|64|32|16|8|4|2|1 -> 128 + 64 + 32 + 16 + 8 + 4 + 1 = 253*
27	01111011	**123**	*128|64|32|16|8|4|2|1 -> 64 + 32 + 16 + 8 + 2 + 1 = 123*
28	00010000	**16**	*128|64|32|16|8|4|2|1 -> 16 = 16*
29	10001010	**138**	*128|64|32|16|8|4|2|1 -> 128 + 8 + 2 = 138*
30	00111011	**59**	*128|64|32|16|8|4|2|1 -> 32 + 16 + 8 + 2 + 1 = 59*

GCSE CS

CodeBash

Number Bases

codebash.co.uk

SOLUTIONS

SOLUTIONS

Section B: Denary to Binary (30 marks)

Convert each denary number to its 8-bit binary equivalent.

Method - Denary to Binary:
Find the largest power of 2 that fits, mark 1. Repeat with remainder.
Column values: 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1
Example: 178 = 128+32+16+2 -> 10110010

Q	Denary	Binary (8-bit)	Working
31	56	00111000	*56 = 32 + 16 + 8 -> 00111000*
32	232	11101000	*232 = 128 + 64 + 32 + 8 -> 11101000*
33	178	10110010	*178 = 128 + 32 + 16 + 2 -> 10110010*
34	237	11101101	*237 = 128 + 64 + 32 + 8 + 4 + 1 -> 11101101*
35	49	00110001	*49 = 32 + 16 + 1 -> 00110001*
36	4	00000100	*4 = 4 -> 00000100*
37	77	01001101	*77 = 64 + 8 + 4 + 1 -> 01001101*
38	28	00011100	*28 = 16 + 8 + 4 -> 00011100*
39	62	00111110	*62 = 32 + 16 + 8 + 4 + 2 -> 00111110*
40	143	10001111	*143 = 128 + 8 + 4 + 2 + 1 -> 10001111*
41	69	01000101	*69 = 64 + 4 + 1 -> 01000101*
42	242	11110010	*242 = 128 + 64 + 32 + 16 + 2 -> 11110010*
43	47	00101111	*47 = 32 + 8 + 4 + 2 + 1 -> 00101111*
44	168	10101000	*168 = 128 + 32 + 8 -> 10101000*
45	193	11000001	*193 = 128 + 64 + 1 -> 11000001*
46	124	01111100	*124 = 64 + 32 + 16 + 8 + 4 -> 01111100*
47	171	10101011	*171 = 128 + 32 + 8 + 2 + 1 -> 10101011*
48	184	10111000	*184 = 128 + 32 + 16 + 8 -> 10111000*
49	8	00001000	*8 = 8 -> 00001000*
50	232	11101000	*232 = 128 + 64 + 32 + 8 -> 11101000*
51	148	10010100	*148 = 128 + 16 + 4 -> 10010100*
52	229	11100101	*229 = 128 + 64 + 32 + 4 + 1 -> 11100101*
53	20	00010100	*20 = 16 + 4 -> 00010100*
54	195	11000011	*195 = 128 + 64 + 2 + 1 -> 11000011*
55	165	10100101	*165 = 128 + 32 + 4 + 1 -> 10100101*

56	185	**10111001**	*185 = 128 + 32 + 16 + 8 + 1 -> 10111001*
57	105	**01101001**	*105 = 64 + 32 + 8 + 1 -> 01101001*
58	129	**10000001**	*129 = 128 + 1 -> 10000001*
59	138	**10001010**	*138 = 128 + 8 + 2 -> 10001010*
60	214	**11010110**	*214 = 128 + 64 + 16 + 4 + 2 -> 11010110*

Number Bases

Section C: Hexadecimal to Denary (20 marks)

Convert each hexadecimal value to its denary equivalent.

Method - Hex to Denary:
Split into two nibbles. Multiply high nibble by 16, add low nibble.
A=10, B=11, C=12, D=13, E=14, F=15
Example: 0xB4 = (11x16)+4 = 176+4 = 180

Q	Hex	Denary	Working
61	`0xC3`	**195**	*0xC3: (12 x 16) + 3 = 192 + 3 = 195*
62	`0xFD`	**253**	*0xFD: (15 x 16) + 13 = 240 + 13 = 253*
63	`0xC8`	**200**	*0xC8: (12 x 16) + 8 = 192 + 8 = 200*
64	`0x8F`	**143**	*0x8F: (8 x 16) + 15 = 128 + 15 = 143*
65	`0xAF`	**175**	*0xAF: (10 x 16) + 15 = 160 + 15 = 175*
66	`0xBF`	**191**	*0xBF: (11 x 16) + 15 = 176 + 15 = 191*
67	`0x4C`	**76**	*0x4C: (4 x 16) + 12 = 64 + 12 = 76*
68	`0x7D`	**125**	*0x7D: (7 x 16) + 13 = 112 + 13 = 125*
69	`0x19`	**25**	*0x19: (1 x 16) + 9 = 16 + 9 = 25*
70	`0xC4`	**196**	*0xC4: (12 x 16) + 4 = 192 + 4 = 196*
71	`0x96`	**150**	*0x96: (9 x 16) + 6 = 144 + 6 = 150*
72	`0x50`	**80**	*0x50: (5 x 16) + 0 = 80 + 0 = 80*
73	`0x89`	**137**	*0x89: (8 x 16) + 9 = 128 + 9 = 137*
74	`0x59`	**89**	*0x59: (5 x 16) + 9 = 80 + 9 = 89*
75	`0xE5`	**229**	*0xE5: (14 x 16) + 5 = 224 + 5 = 229*
76	`0x90`	**144**	*0x90: (9 x 16) + 0 = 144 + 0 = 144*
77	`0xE2`	**226**	*0xE2: (14 x 16) + 2 = 224 + 2 = 226*
78	`0x61`	**97**	*0x61: (6 x 16) + 1 = 96 + 1 = 97*
79	`0x31`	**49**	*0x31: (3 x 16) + 1 = 48 + 1 = 49*
80	`0x08`	**8**	*0x08: (0 x 16) + 8 = 0 + 8 = 8*

Number Bases

Section D: Denary to Hexadecimal (20 marks)

Convert each denary value to two-digit hexadecimal.

Method - Denary to Hex:
Divide by 16: quotient = high nibble, remainder = low nibble.
Values 10-15 -> A-F
Example: 180 / 16 = 11 r 4 -> B4 -> 0xB4

Q	Denary	Hex	Working
81	106	0x6A	*106: 106//16=6 106%16=10 -> 0x6A*
82	113	0x71	*113: 113//16=7 113%16=1 -> 0x71*
83	182	0xB6	*182: 182//16=11 182%16=6 -> 0xB6*
84	11	0x0B	*11: 11//16=0 11%16=11 -> 0x0B*
85	108	0x6C	*108: 108//16=6 108%16=12 -> 0x6C*
86	226	0xE2	*226: 226//16=14 226%16=2 -> 0xE2*
87	248	0xF8	*248: 248//16=15 248%16=8 -> 0xF8*
88	70	0x46	*70: 70//16=4 70%16=6 -> 0x46*
89	4	0x04	*4: 4//16=0 4%16=4 -> 0x04*
90	52	0x34	*52: 52//16=3 52%16=4 -> 0x34*
91	106	0x6A	*106: 106//16=6 106%16=10 -> 0x6A*
92	36	0x24	*36: 36//16=2 36%16=4 -> 0x24*
93	216	0xD8	*216: 216//16=13 216%16=8 -> 0xD8*
94	183	0xB7	*183: 183//16=11 183%16=7 -> 0xB7*
95	243	0xF3	*243: 243//16=15 243%16=3 -> 0xF3*
96	64	0x40	*64: 64//16=4 64%16=0 -> 0x40*
97	130	0x82	*130: 130//16=8 130%16=2 -> 0x82*
98	4	0x04	*4: 4//16=0 4%16=4 -> 0x04*
99	65	0x41	*65: 65//16=4 65%16=1 -> 0x41*
100	171	0xAB	*171: 171//16=10 171%16=11 -> 0xAB*

Number Bases

Section E: Binary and Hexadecimal Conversion (20 marks)

E1: Binary to Hex. E2: Hex to Binary.

Method - Binary-Hex Conversion:
Split 8-bit binary into two groups of 4 (nibbles). Convert each nibble.
0000=0, 0001=1 ... 1010=A, 1011=B, 1100=C, 1101=D, 1110=E, 1111=F
Example: 10110100 -> 1011|0100 -> B|4 -> 0xB4

E1: Binary to Hexadecimal (Q101-Q110)

Q	Binary	Hex	Working
101	`10011000`	**0x98**	*1001|1000 -> 9|8 -> 0x98*
102	`10100001`	**0xA1**	*1010|0001 -> A|1 -> 0xA1*
103	`01010010`	**0x52**	*0101|0010 -> 5|2 -> 0x52*
104	`00011100`	**0x1C**	*0001|1100 -> 1|C -> 0x1C*
105	`10110011`	**0xB3**	*1011|0011 -> B|3 -> 0xB3*
106	`01000101`	**0x45**	*0100|0101 -> 4|5 -> 0x45*
107	`01011000`	**0x58**	*0101|1000 -> 5|8 -> 0x58*
108	`11101101`	**0xED**	*1110|1101 -> E|D -> 0xED*
109	`00110010`	**0x32**	*0011|0010 -> 3|2 -> 0x32*
110	`01001110`	**0x4E**	*0100|1110 -> 4|E -> 0x4E*

E2: Hexadecimal to Binary (Q111-Q120)

Q	Hex	Binary (8-bit)	Working
111	`0x97`	**10010111**	*0x9|0x7 -> 1001|0111 -> 10010111*
112	`0x09`	**00001001**	*0x0|0x9 -> 0000|1001 -> 00001001*
113	`0xC6`	**11000110**	*0xC|0x6 -> 1100|0110 -> 11000110*
114	`0x2F`	**00101111**	*0x2|0xF -> 0010|1111 -> 00101111*
115	`0x7E`	**01111110**	*0x7|0xE -> 0111|1110 -> 01111110*
116	`0x70`	**01110000**	*0x7|0x0 -> 0111|0000 -> 01110000*
117	`0x98`	**10011000**	*0x9|0x8 -> 1001|1000 -> 10011000*
118	`0x4B`	**01001011**	*0x4|0xB -> 0100|1011 -> 01001011*
119	`0x93`	**10010011**	*0x9|0x3 -> 1001|0011 -> 10010011*
120	`0xD5`	**11010101**	*0xD|0x5 -> 1101|0101 -> 11010101*

Number Bases

Section F: Two's Complement - Read the Value (25 marks)

Each 8-bit two's complement binary represents a negative value. State the denary.

Method - Two's Complement (reading):
If MSB (leftmost bit) = 1, the number is negative.
Method A: Unsigned value - 256 = signed value
Method B: Invert all bits, add 1, negate
Example: 11010110 -> 214 - 256 = -42

Q	Binary (two's comp)	Denary	Working
121	`11000100`	**-60**	*MSB=1 (negative): 196 - 256 = -60*
122	`11101100`	**-20**	*MSB=1 (negative): 236 - 256 = -20*
123	`10000001`	**-127**	*MSB=1 (negative): 129 - 256 = -127*
124	`10001010`	**-118**	*MSB=1 (negative): 138 - 256 = -118*
125	`10110000`	**-80**	*MSB=1 (negative): 176 - 256 = -80*
126	`11000010`	**-62**	*MSB=1 (negative): 194 - 256 = -62*
127	`11110000`	**-16**	*MSB=1 (negative): 240 - 256 = -16*
128	`11110101`	**-11**	*MSB=1 (negative): 245 - 256 = -11*
129	`11000010`	**-62**	*MSB=1 (negative): 194 - 256 = -62*
130	`10010111`	**-105**	*MSB=1 (negative): 151 - 256 = -105*
131	`11100101`	**-27**	*MSB=1 (negative): 229 - 256 = -27*
132	`10001010`	**-118**	*MSB=1 (negative): 138 - 256 = -118*
133	`11111011`	**-5**	*MSB=1 (negative): 251 - 256 = -5*
134	`10011011`	**-101**	*MSB=1 (negative): 155 - 256 = -101*
135	`10000101`	**-123**	*MSB=1 (negative): 133 - 256 = -123*
136	`11000101`	**-59**	*MSB=1 (negative): 197 - 256 = -59*
137	`11011110`	**-34**	*MSB=1 (negative): 222 - 256 = -34*
138	`10101111`	**-81**	*MSB=1 (negative): 175 - 256 = -81*
139	`10111111`	**-65**	*MSB=1 (negative): 191 - 256 = -65*
140	`10111010`	**-70**	*MSB=1 (negative): 186 - 256 = -70*
141	`10000001`	**-127**	*MSB=1 (negative): 129 - 256 = -127*
142	`10000101`	**-123**	*MSB=1 (negative): 133 - 256 = -123*
143	`10011101`	**-99**	*MSB=1 (negative): 157 - 256 = -99*
144	`11011111`	**-33**	*MSB=1 (negative): 223 - 256 = -33*

145	11000000	**-64**	*MSB=1 (negative): 192 - 256 = -64*

Number Bases

Section G: Two's Complement - Convert Negative Denary (15 marks)

Convert each negative denary value to 8-bit two's complement binary.

Method - Two's Complement (converting):
Step 1: Take the magnitude (ignore the minus sign) and write in binary.
Step 2: Invert all bits (0->1, 1->0).
Step 3: Add 1 to the result.
Example: -42 -> 00101010 -> 11010101 -> +1 -> 11010110

Q	Negative Denary	Binary (two's comp)	Working
146	-28	11100100	*Step 1: +28 in binary = 00011100 Step 2: Invert = 11100011 Step 3: Add 1*
147	-68	10111100	*Step 1: +68 in binary = 01000100 Step 2: Invert = 10111011 Step 3: Add 1*
148	-53	11001011	*Step 1: +53 in binary = 00110101 Step 2: Invert = 11001010 Step 3: Add 1*
149	-16	11110000	*Step 1: +16 in binary = 00010000 Step 2: Invert = 11101111 Step 3: Add 1*
150	-62	11000010	*Step 1: +62 in binary = 00111110 Step 2: Invert = 11000001 Step 3: Add 1*
151	-82	10101110	*Step 1: +82 in binary = 01010010 Step 2: Invert = 10101101 Step 3: Add 1*
152	-83	10101101	*Step 1: +83 in binary = 01010011 Step 2: Invert = 10101100 Step 3: Add 1*
153	-27	11100101	*Step 1: +27 in binary = 00011011 Step 2: Invert = 11100100 Step 3: Add 1*
154	-25	11100111	*Step 1: +25 in binary = 00011001 Step 2: Invert = 11100110 Step 3: Add 1*
155	-75	10110101	*Step 1: +75 in binary = 01001011 Step 2: Invert = 10110100 Step 3: Add 1*
156	-12	11110100	*Step 1: +12 in binary = 00001100 Step 2: Invert = 11110011 Step 3: Add 1*
157	-7	11111001	*Step 1: +7 in binary = 00000111 Step 2: Invert = 11111000 Step 3: Add 1 =*
158	-106	10010110	*Step 1: +106 in binary = 01101010 Step 2: Invert = 10010101 Step 3: Add 1*
159	-2	11111110	*Step 1: +2 in binary = 00000010 Step 2: Invert = 11111101 Step 3: Add 1 =*
160	-120	10001000	*Step 1: +120 in binary = 01111000 Step 2: Invert = 10000111 Step 3: Add 1*

Number Bases

Section H: Binary Addition (20 marks)

Add the two 8-bit binary numbers. State the 8-bit result, denary total, and overflow status.

Method - Binary Addition:
Add column by column from right to left, carrying into the next column.
0+0=0, 0+1=1, 1+1=0 carry 1, 1+1+1=1 carry 1
Overflow occurs if the true total exceeds 255 (cannot fit in 8 bits).

Q	Operand A	Operand B	Result (8-bit)	Denary	Overflow?
161	00010101	00111011	01010000	80	No
162	01101001	00000011	01101100	108	No
163	00001111	00011100	00101011	43	No
164	00101011	00000001	00101100	44	No
165	01110010	00101110	10100000	160	No
166	00111000	01011011	10010011	147	No
167	00000011	01100100	01100111	103	No
168	00000001	01110111	01111000	120	No
169	01111111	01010001	11010000	208	No
170	01101110	01100111	11010101	213	No
171	01111100	01110011	11101111	239	No
172	00101100	00011011	01000111	71	No
173	00101000	01000001	01101001	105	No
174	01011000	01111110	11010110	214	No
175	00000111	00110111	00111110	62	No
176	01001101	01111001	11000110	198	No
177	00010100	01101010	01111110	126	No
178	00011000	00110100	01001100	76	No
179	01100001	00111011	10011100	156	No
180	00011111	00110111	01010110	86	No

Number Bases

Section I: Logical Bit Shifts (20 marks)

Apply the stated logical bit shift. Give the result in binary and state the new denary value.

Method - Logical Bit Shifts:
Left shift by 1: multiply by 2 (shift all bits left, fill right with 0).
Right shift by 1: integer divide by 2 (shift all bits right, fill left with 0).
Bits shifted off the end are lost. Left shift by 2 multiplies by 4, etc.

Q	Value (binary)	Shift	Result (binary)	Result (denary)	Working
181	00001001	1 left	00010010	18	*x2=18*
182	00001100	1 left	00011000	24	*x2=24*
183	00100111	1 right	00010011	19	*/2=19*
184	00110100	1 right	00011010	26	*/2=26*
185	00010111	2 left	01011100	92	*x4=92*
186	00110011	1 left	01100110	102	*x2=102*
187	00100101	1 left	01001010	74	*x2=74*
188	00110111	2 right	00001101	13	*/4=13*
189	00001001	2 right	00000010	2	*/4=2*
190	00001011	2 left	00101100	44	*x4=44*
191	00110010	1 right	00011001	25	*/2=25*
192	00011001	1 left	00110010	50	*x2=50*
193	00111100	1 left	01111000	120	*x2=120*
194	00010010	1 right	00001001	9	*/2=9*
195	00100000	1 left	01000000	64	*x2=64*
196	00010011	2 left	01001100	76	*x4=76*
197	00100000	1 left	01000000	64	*x2=64*
198	00101001	2 right	00001010	10	*/4=10*
199	00100110	1 right	00010011	19	*/2=19*
200	00000011	1 left	00000110	6	*x2=6*

Want more like this?

CodeBash has the rest of the GCSE and A-Level CS toolkit.

More Worksheets & Mark Schemes
Logic Gates, Karnaugh Maps, Floating Point, SQL and trace tables for OCR, AQA, CIE and Edexcel.

Online Code Editor
Write and run Python and C# in the browser. No setup. Save your work to your account.

Lockdown Assessments
Timed, exam-style assessments your teacher can release to the whole class with one click.

Theory Revision
Spaced flashcards, quizzes and explanations covering the full GCSE and A-Level specification.

codebash.co.uk

Free for students. School plans available for teachers.

Made for UK Computer Science classrooms by **CodeBash**. Share this booklet with anyone who would find it useful.