# 1. ASCII

1. F

I know that the value of F is 70. Therefore, it uses 8 bits and in order to figure out which ones are 1’s and 0’s I would have to see if I could subtract the value from whichever decimal of the base 2’s, that can be subtracted. The values on the left are base 2’s which are equal to base 10 decimals i.e. 1, 2, 4, 8 … etc.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 27 = 128 | 26 = 64 | 25 = 32 | 24 = 16 | 23 = 8 | 22 = 4 | 21 = 2 | 20 = 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |

01000110

1. m

I know that the value of m is 109. Therefore, it uses 8 bits and in order to figure out which ones are 1’s and 0’s I would have to see if I could subtract the value from whichever decimal of the base 2’s, that can be subtracted. The values on the left are base 2’s which are equal to base 10 decimals i.e. 1, 2, 4, 8 … etc.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 27 = 128 | 26 = 64 | 25 = 32 | 24 = 16 | 23 = 8 | 22 = 4 | 21 = 2 | 20 = 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |

01101101

1. €

I know that the value of € is 128. Therefore, it uses 8 bits and in order to figure out which ones are 1’s and 0’s I would have to see if I could subtract the value from whichever decimal of the base 2’s, that can be subtracted. The values on the left are base 2’s which are equal to base 10 decimals i.e. 1, 2, 4, 8 … etc.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 27 = 128 | 26 = 64 | 25 = 32 | 24 = 16 | 23 = 8 | 22 = 4 | 21 = 2 | 20 = 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

10000000

1. 8

I know that the value of 8 is 56. Therefore, it uses 8 bits and in order to figure out which ones are 1’s and 0’s I would have to see if I could subtract the value from whichever decimal of the base 2’s, that can be subtracted. The values on the left are base 2’s which are equal to base 10 decimals i.e. 1, 2, 4, 8 … etc.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 27 = 128 | 26 = 64 | 25 = 32 | 24 = 16 | 23 = 8 | 22 = 4 | 21 = 2 | 20 = 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

00111000

1. ©

I know that the value of © is 169. Therefore, it uses 8 bits and in order to figure out which ones are 1’s and 0’s I would have to see if I could subtract the value from whichever decimal of the base 2’s, that can be subtracted. The values on the left are base 2’s which are equal to base 10 decimals i.e. 1, 2, 4, 8 … etc.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 27 = 128 | 26 = 64 | 25 = 32 | 24 = 16 | 23 = 8 | 22 = 4 | 21 = 2 | 20 = 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |

10101001

# 2. Convert decimal to binary

1. 29

29-16 = 8 – 13 = 5 – 4 = 1- 1 = 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 27 = 128 | 26 = 64 | 25 = 32 | 24 = 16 | 23 = 8 | 22 = 4 | 21 = 2 | 20 = 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |

00011101

1. 92

92-64 = 28 – 16 = 12 – 8 = 4 – 4 = 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 27 = 128 | 26 = 64 | 25 = 32 | 24 = 16 | 23 = 8 | 22 = 4 | 21 = 2 | 20 = 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |

01011100

1. 156

156 – 128 = 28 – 16 = 12 – 8 = 4 – 4 = 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 27 = 128 | 26 = 64 | 25 = 32 | 24 = 16 | 23 = 8 | 22 = 4 | 21 = 2 | 20 = 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

10011100

1. 203

203 – 128 = 75 -64 = 11 – 8 = 3 – 2 = 1 – 1 = 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 27 = 128 | 26 = 64 | 25 = 32 | 24 = 16 | 23 = 8 | 22 = 4 | 21 = 2 | 20 = 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |

11001011

1. 233

233 – 128 = 105 – 64 = 41 – 32 = 9 – 8 = 1 -1 = 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 27 = 128 | 26 = 64 | 25 = 32 | 24 = 16 | 23 = 8 | 22 = 4 | 21 = 2 | 20 = 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |

11101001

# 3. Robot Instructions

1. We would need only four bits because our decimal number is only 13. In base 2 the first four bits all add up to 15 as of base 10. It would make no point in adding more bits because we will never utilize them since the binary of 13 is 1101.
2. I will assume that the two extensions are just to fill the last of the remaining decimal/binary field of only 4 columns of base 2.

|  |  |  |
| --- | --- | --- |
| Instructions | Binary | Decimal |
| Raise arm one unit of distance | 0001 | 1 |
| Lower arm one unit of distance | 0010 | 2 |
| Rotate arm clockwise one degree | 0011 | 3 |
| Rotate arm counterclockwise one degree | 0100 | 4 |
| Extend arm one unit of distance | 0101 | 5 |
| Retract arm one unit of distance | 0110 | 6 |
| Grip | 0111 | 7 |
| Release Grip | 1000 | 8 |
| Rotate grip clockwise one degree | 1001 | 9 |
| Rotate grip counterclockwise one degree | 1010 | 10 |
| Power On | 1011 | 11 |
| Power Off | 1100 | 12 |
| Self Test | 1101 | 13 |
| Extension 1 | 1110 | 14 |
| Extension 2 | 1111 | 15 |

d. Like I said in “a” that I only need 4 bits because the first four bits of base 2 all add up to only 15 so I can accommodate two extra instructions.

Calculations:

20 = 1, 21 = 2, 22 = 4, 23 = 8. To get the binary all we got to do is subtract the decimal number from the base 10’s that each 2 columns that can be subtracted.

e. Since we must add 9 new instructions that would mean that I would have to increase my bits to 5 bits because 4 bits can only handle up to 15, base 10 decimals. Just like the calculation above all the bits would add up to 31 in decimal form of base 10. Then just like how I add the bits I would just add it the same way since there are now 5 columns of base 2. All we got to do is convert each column to their decimal form and we can start matching instructions again like before.

f.

|  |  |  |
| --- | --- | --- |
| Instructions | Binary | Decimal |
| Raise arm one unit of distance | 00001 | 1 |
| Lower arm one unit of distance | 00010 | 2 |
| Rotate arm clockwise one degree | 00011 | 3 |
| Rotate arm counterclockwise one degree | 00100 | 4 |
| Extend arm one unit of distance | 00101 | 5 |
| Retract arm one unit of distance | 00110 | 6 |
| Grip | 00111 | 7 |
| Release Grip | 01000 | 8 |
| Rotate grip clockwise one degree | 01001 | 9 |
| Rotate grip counterclockwise one degree | 01010 | 10 |
| Power On | 01011 | 11 |
| Power Off | 01100 | 12 |
| Self Test | 01101 | 13 |
| Extension 1 | 01110 | 14 |
| Extension 2 | 01111 | 15 |
| Improvement 1 | 01101 | 16 |
| Improvement 2 | 10001 | 17 |
| Improvement 3 | 10010 | 18 |
| Improvement 4 | 10011 | 19 |
| Improvement 5 | 10100 | 20 |
| Improvement 6 | 10101 | 21 |
| Improvement 7 | 10110 | 22 |
| Improvement 8 | 10111 | 23 |
| Improvement 9 | 11000 | 24 |