1) Assembly language isn't portable as it's specific to the architecture to whatever system you're writing it for. It is also very close to machine language in that there is a 1 to 1 relationship between assembly language instructions and machine language instructions, which would make it extremely long to write and difficult to maintain large programs.

ASSN 2

2) A 11110111
B + 10110110
110100001

∴ 1 10100001

3) −5 → 101 → 010 → 011
−2 → 010 → 101 → 110
+4 → 100 → 100 → 100
−3 1101 → 0010 → 0011 = 3 ∴ −3

4) 4 5 7 8 9 8 A 0
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
0100 0101 0111 1000 1001 1000 1010 0000
∴ 10001010111110001001100010100000

5) −39 39/16 → 2 R7
2/16 → 0 R2
⇒ 0027 FFD8 + 1 = FFD9

6) a) 10010101 ⇒ 01101010 + 1 = 01101011
$2^6+2^5+2^3+2^1+2^0$ = 107 ∴ −107

b) 01110111
$2^6+2^5+2^4+2^2+2^1+2^0$ = 119

7. ABC + DEF = 18AB

	ABC
+	DEF
	18AB
−	0F51

12 + 15 = 27; 27/16 = 1 R 11

11 + 14 + 1 = 26; 26/16 = 1 R 10

10 + 13 + 1 = 24; 24/16 = 1 R 8

⇒ 18AB + F0AF = X095A

11 + 15 = 26; 26/16 = 1 R 10

10 + 10 + 1 = 21; 21/16 = 1 R 5

∴ **95A**

8) −11 ⇒ 11/2 = 5 R1

5/2 = 2 R1

2/2 = 1 R0

1/2 = 0 R1

1011 ⇒ 0100 + 1 = 0101

unable to store in 4-bit word require 5th bit for sign bit

also because 4-bit 2s comp only store values between

-2^{n-1} – $+2^{n-1}-1$

-2^{3} – $+2^{3}-1$

−8 – +7 which excludes −11

9) $-2^{n-1} = -2^{182}$

10) $(\neg x \wedge \neg y \wedge z) \vee (\neg x \wedge y \wedge \neg z) \vee (x \wedge \neg y \wedge \neg z)$