# Exercise 2 Answers

1.

* 1. F6 20 00

CB 0A

7B 20 00

* 1. CD 30 00

EC 40

* 1. 96 20

C6 14

18 06

5A 21

* 1. B6 25 00

B1 25 01

20 50

* 1. DC 40

83 00 01

20 9C

1. In assembly the offset is always written as a sign and positive magnitude even when using hexadecimal, but the machine code always uses the 2’s complement format. For example, an offset of $10 would always be interpreted as +1610 and result in a 9-bit offset, since there is now way to know that $10 was supposed to represent only a 5-bit offset of -1610.
2. LDAA 5,X A6 05
3. LDAA -$10,X A6 10
4. LDAB -$20,X E6 E1 E0
5. LDAB -$9B,x E6 E1 65
6. LDX -$F0,X EE E1 10
7. LDD -$46,X EC E1 BA
8. LDD $63,X EC E0 63
9. LDY $71,X ED E0 71
10. LDS $10,X EF E0 10
11. LDS -$BE,X EF E1 42

3.

1. LDAA #$45
2. BRA $21
3. INC $1900
4. ADDA $CC,Y
5. SUBD #$2000
6. DEC 0,X
7. DEX
8. INY
9. LBEQ $1000
10. ABA

4.

1. $4A
2. $7F
3. $C7
4. $D3
5. $C100
6. $F044
7. $C1C2
8. $B124
9. $3555
10. $1B
11. $0F
12. $7D
13. $29
14. $BFFF
15. $CFBA
16. $C0BE
17. $B0D8
18. $3357

5.

1. $CFAC
2. $81EF
3. $2D6A
4. $C000
5. $E044
6. $C0C2
7. $B124
8. $3455

6.

1. $20A0
2. $2164
3. $3FBA
4. $1EAB
5. $2D89
6. $2697
7. The difference between this and the previous problem is that a short branch always (BRA) is two bytes long, so the offset is applied to an address two bytes further (i.e. higher) in memory. A long branch always (LBRA) is four bytes long, and the offset is applied to an address four bytes higher in memory than the start of the branch.
8. $20A2
9. $2166
10. $3FBC
11. $1EAD
12. $2D8D
13. $269B
14. The offset is calculated from the memory address after the branch instruction. A BRA is 2 bytes long, occupying addresses 2450h and 2451h, and the offsets are calculated from 2452h. The LBRA instruction is 4 bytes long, and the offsets are calculated from 2454h.
    1. BRA $0E
    2. BRA $EE
    3. LBRA $00AF
    4. BRA $B1
    5. LBRA $0BAC
    6. LBRA $EBAC
    7. BRA $5C
    8. BRA $A5
    9. LBRA $1131
    10. LBRA $EDE0