1. The byte 33(16) is stored in memory address 101(16).
2. 3.1

A \* 17

A (16 + 1)

Let A be in R1

ADD R0, R1, R1, LSL#4 ; R0 := A + 16A

3.2

A \* 23

A (16 + 7)

Let A be in R1

MOV R0, R1, LSL #4 ; R0 := 16A

RSB R2, R1, R1, LSL #3 ; R2 := 8A – A

ADD R0, R0, R2 ; R0 := 16A + 7A

1. A \* 105

A (15 \* 7)

RSB R0, R0, R0, LSL #3 ; R0 := 8A - A

RSB R0, R0, R0, LSL #4 ; R0 := (7A \* 16) - 7A

1. If (a == b) and (c == d)

Then d = a + b + c + 1

Let a be in R1, b be in R2, c be in R3, d be in R4

TEQ R1, R2 ; test a == b

BNE DONE

TEQ R3, R4 ; test c == d

BNE DONE

ADD R4, R1, R2 ; R4 := a + b

ADD R4, R3, #1 ; R4 := c + 1

DONE

1. If a < 0 then a = 0 – a

Let a be in R1

CMP R1, #0 ; a < 0

BGE DONE

SUB R1, #0, R1 ; R1 := 0 - a

1. R1 = I

R2 = c

A = #65

CMP R1, #0 ; I < 0

BGE ELSE

ADD R2, R1, #48 ; R2 := I + 48

ELSE ADD R2, R1, #55 ; R2 := I + 55

1. R2 = 0
2. Move value of 0 into r2
3. Compare r0 and r0. If r0 is less than r0, then go to the Label Equal. However, we see that r0 equals r0, so we go ahead to the unconditional branch less.
4. Load the value 0xABCD into r0 and then go to done

The value of r2 wasn’t changed since it was loaded with the value 0.

1. R2 = 2
2. Load the value 0 into r2
3. Compare r0 and r1. If r0 <= r1 go to the label equal.
4. r0 is less than r1 so go the label Equal
5. Load the value 2 into r2
6. Unconditional branch to OK
7. R2 = 1111 1111 1111 1111 1111 1111 1111 1101(2) = -3(10)
8. Load the value of 0 into r2
9. Compare r0 and r1(comparison by unsigned value because of operand BHS)
10. r0 is higher than r1 so go to branch label Equal
11. load the inverse bitwise representation of the value 2
    1. 32 bit wise representation of 2
       1. 0000 0000 0000 0000 0000 0000 0000 0010(2) = 2(10)
    2. Inverse representation of 2
       1. 1111 1111 1111 1111 1111 1111 1111 1101(2) = -3(10)
12. Unconditional branch to OK
13. Write the following program. Load a constant 0xABCDEF in register R5. Perform swap between 1st and 4th byte, and also between 2nd and 3rd byte. Load the result in R6. After that perform reverse swap again between 1st and 4th byte, and also between 2nd and 3rd byte. At the end compare R5 and R6, they have to be equal.

0xABCDEF

1st byte AB, 2nd byte CD, 3rd byte EF

LDR R5, #0X00ABCDEF ; R5 := 0x00ABCDEF

AND R1, R5, #0x000000FF ; R1 := 0x00ABCDEF AND 0X000000FF

ORR R6, R5, R1, LSL# 26 ; R6 := 0X00ABCDEF ORR 0XEF000000

AND R6, R6, #0XFFFFFF00 ; R6 := 0XEFABCDEF AND 0XFFFFFF00

AND R2, R5, #0X00FF0000 ; R2 := 0X00ABCDEF AND 0X00FF0000

AND R3, R5, #0X0000FF00 ; R3 := 0X00ABCDEF AND 0X0000FF00

ORR R6, R6, R2, LSR #8 ; R6 := 0XEFABCD00 ORR 0X0000AB00

ORR R6, R6, R3, LSL #8 ; R6 := 0XEFABAB00 ORR 0X00CD0000

ORR R6, R6, R2, LSL #8 ; R6 := 0XEFCDAB00 ORR 0X00AB0000

ORR R6, R6, R3, LSR #8 ; R6 := 0XEFABAB00 ORR 0X0000CD00

ORR R6, R6, R1, LSR #26 ; R6 := 0XEFABCD00 ORR 0X000000EF

AND R6, R6, #0X00FFFFFF ; R6 := 0XEFABCDEF AND 0X00FFFFFF