# Worksheet 4 Assembly language Answers

**Task 1**

1. A positive number is held in memory location 201. Write assembly code instructions which put the negative of the number in location 202.

 MOV R1, #0 ;initialise R1 to zero

 LDR R2, 201 ;load value from location 201 into R2

 SUB R1, R1, R2 ;subtract value from R1 (which is zero)

 STR R1, 202 ;store result in location 202

2. Write assembly code instructions to take three numbers held in locations 201, 202 and 203 and store them back in the same locations in reverse order.

 LDR R1, 201 ;load first number into R1

 LDR R3, 203 ;load third number into R3

 STR R1, 203 ;store first number in third location

 STR R3, 201 ;store third number in first location

Could use any two registers. R1 and R3 used for easier reference.

3. Three numbers are held in locations 201, 202 and 203. Write assembly code instructions to store the maximum of the three numbers in location 300.

 LDR R1, 201 ;load first number into R1

 LDR R2, 202 ;load second number into R2

 CMP R1, R2 ;compare value in R1 with second number

 BGT label1 ;branch if R1 > R2 label1

 MOV R1, R2 ;store larger number in R1

 label1:

 LDR R3, 203 ;load the third number

 CMP R1, R3 ;compare value in R1 with third number

 BGT label2 ;branch if R1 > R3 to label2

 MOV R1, R3 ;store larger number in R1

 label2:

 STR R1, 300 ; store largest number in 300

4. Assume that numbers can be input by a user into Register 0 using the instruction

INP R0

 (a) Ten numbers are input by the user. Write assembly code instructions to add the ten numbers as they are input and store the result in location 300.

 MOV R1, #0 ;initialise R1 to 0 to hold total

 MOV R2, #0 ;initialise R2 to hold count

 LOOP

 INP R0 ;input a number

 ADD R1, R1, R0 ;add number to total

 ADD R2, R2, #1 ;increment count

 CMP R2, #10 ;have 10 numbers been input?

 BNE LOOP ;if not, branch to LOOP

 STR R1, 300 ;if yes, store total in location 300

 (b) Ten integers are input by the user. Write assembly code instructions to count the number of integers that are greater than or equal to 30. The result should be stored in location 301.

 MOV R1, #0 ;initialise R1 to 0 to hold total>=30

 MOV R2, #0 ;initialise R2 to hold count

 LOOP

 INP R0 ;input a number

 ADD R2, R2, #1 ;increment count

 CMP R0, #30 ;is the number in R0 < 30?

 BLT label2 ;yes, so skip the next instruction

 ADD R1, R1, #1 ;no, so add 1 to total of numbers >= 30

 label2:

 CMP R2, #10 ;have 10 numbers been input?

 BNE LOOP ;if not, branch to LOOP

 STR R1, 301

 (continue)

**Task 2**

1. Write assembly code instructions to find whether a number held in location 200 is odd or even. If the number is even, branch to label1, otherwise continue with the next statement.

 LDR R1, 200 ;load the number from 200 into R1

 AND R2, R1, #1 ;AND the number with 00000001

 CMP R2, #0 ;compare the result of the AND with zero

 BEQ label1 ;if the number was even, R2 will be zero

 (continue)

label1:

 (continue)

2. A bit pattern held in R0 represents 8 switches numbered 1 to 8 (left to right).

 (a) Write assembly code instructions to initialise the switches to zero, and then turn on switches 1,3,5 and 7.

 MOV R0, #0 ;move 0 to R0 to initialise switches

 ORR R0, R0, #10101010B ;Turn on switches 1,3,5,7, store in R0

 ;where B indicates a binary value

 (b) Assume you do not know the state of the switches in R0. Write an assembly code instruction to turn off any switches that are on, and vice versa.

 MVN R0, R0; perform NOT operation on R0

or could use EOR:

 EOR R0, R0, #11111111B

3. What will be the effect of performing an XOR operation on an operand with itself?

 Give an example.

 The result is zero.

 For example: A 0101 1111

 A 0101 1111

 After XOR: A 0000 0000

**Task 3**

1. Assume R0 contains an 8-bit positive integer. Using logical shifts, compare and branch operations, write assembly code statements to branch to label1 if the integer represents an even number, otherwise continue to the next statement.

 Test your program by tracing the contents of R0 and any other registers used.

 MOV R1, R0 ;copy R0 into R1

 LSR R1, R1, #1 ;shift right one bit

 LSL R1, R1, #1 ;shift left one bit

 CMP R1, R0 ;compare the contents of R0 and R1

 BEQ label1 ;if the number is even, the registers will be equal

 (continue)

 Trace table:

|  |  |  |
| --- | --- | --- |
| **R0** | **R1** |  |
| 11010011 | 11010011 | Copy contents |
|  | 01101001 | Shift right |
|  | 11010010 | Shift left; R0 and R1 not equal so number in R0 is odd |

 Can you think of another way of testing whether the number is even?

 You could use a mask of 00000001 and a logical OR, putting the result in R1. If the number in R0 is odd, R0 will equal R1. If the number is even, then R0 will not be equal to R1.

 OR R1, R0, #1 ;test the rightmost bit

 CMP R0, R1 ;compare the result with 0

 BNE label1 ;the number is even