```
#Use the MARS simulator to write and execute MIPS assembly code that will perform
#the high level language pseudocode assignment statements.
#Use this file as a base and perform each exercise in order according to the
#descriptions given in each comment.
#No pseudo instructions are allowed, use only the following basic MIPS instructions:
#	add	-	arithmetic add
#	sub	-	arithmetic subtract
#	and	-	bitwise and
#	or	-	bitwise or
#	slt	-	set on less than
#	nor	-	bitwise nor
#	addi	-	add with immediate data
#
#YOUR SUBMISSION WILL BE ASSIGNED A SCORE OF ZERO IF IT CONTAINS:
#	PSEUDOINSTRUCTIONS
#	SYNTAX ERRORS
#	INSTRUCTIONS NOT LISTED ABOVE
##########################################
###########################################
## Example of how to perform exercises ##
###########################################
#Variables A through C have the following register mapping:
#	A:$16	B:$17	C:$18
#For temporary storage use registers $13, $14, or $15 as needed for complex operations.
#Example Exercise:
#	Initial Variable Values: A = 50; B = 100; C = 150;
#	Assignment statement to perform: C = A + B - C;
addi $16, $0, 50#A = 50;
addi $17, $0, 100	#B = 100;
```

```
addi $18, $0, 150          #C = 150;
add $13, $16, $17          #Temp = A + B;
sub $18, $13, $18          #C = A + B - C;
#########################################
##      END         OF      EXAMPLE         SECTION         ##
#########################################
########################################

########################################
#Now for your assignment, complete the following exercises in order as they appear.
#The result of each exercise can depend on the result of previous exercises.
#Variables A through F have the following register mapping:
#       A:$10   B:$11   C:$12   D:$13   E:$14   F:$15   G:$16   H:$17   I:$18
#For temporary storage use register $1 through $9 as needed for complex operations.

#Exercise 1:  Complex Assignment statement
#       Initial Variable Values: G = 10000; C = 12000; D = 5678;
#       Assignment statement to perform: H = G+(C-D)
addi $16, $0, 10000
addi $12, $0, 12000
addi $13, $0, 5678
sub $1, $12, $13
add $17, $16, $1
#Exercise 2:  Sum of differences
#       Initial Variable Values: C = 2000; D = 1250; E = 5000; F = 3500;
#       Assignment statement to perform: I = (C-D)+(E-F)
addi $12, $0, 2000
addi $13, $0, 1250
addi $14, $0, 5000
```

```
addi $15, $0, 3500

sub $1, $12, $13

sub $2, $14, $15

add $18, $1, $2

#Exercise 3:  Add without adding

#	Initial Variable Values: A = 25; B = 50

#	Assignment statement to perform: G = A - (~B + 1);

addi $10, $0, 25

addi $11, $0, 50

nor $1, $0, $11

addi $2, $1, 1

sub $16, $10, $2

#Exercise 4: Subtract without Subtracting

#	Initial Variable Values: C = 1000; D = 750;

#	Assignment statement to perform: F = C + (~D + 1)

addi $13, $0, 1000

addi $14, $0, 750

nor $1, $0, $14

addi $2, $1, 1

add $15, $2, $13

#Exercise 5:  Less Than Comparison

#	Initial Variable Values: A = 20000; B = 30000

#	Assignment statement to perform: E = (A<B) ? 1 : 0; //ternary operator

addi $10, $0, 20000

addi $11, $0, 30000

slt $14, $10, $11

#Exercise 6: Exclusive OR without XOR

#	Initial Variable Values:   B = 0x0F0F;      C = 0x0FFF;

#	Assignment statement to perform: D = (B & ~C) | (~B & C)
```

addi $11, $0, 0x0F0F

addi $12, $0, 0x0FFFf

nor $1, $0, $11

nor $2, $0, $12

and $3, $11, $2

and $4, $1, $12

or $13, $3, $4

Convert the following MIPS assembly code into machine language. Show details, show binary and hexadecimal values

MIPS Assembly	Binary	Hex
add $t0, $s0, $s1	0000 0010 0001 0001 0100 0000 0010 0000	Ox02114020
lw $t0, 0x20($t7)	1000 1101 1110 1000 0000 0000 0010 0000	0x8DE80020
addi $s0, $0, -10	0010 0000 0001 0000 1111 1111 1111 0110	0X2010FFF6
addi $s0, $0, 73	0010 0000 0001 0000 0000 0000 0100 1001	0x20100049
sw $t1, -7($t2)	1010 1101 0100 1001 1111 1111 1111 1001	0XAD49FFF9
sub $t1, $s7, $s2	0000 0010 1111 0010 0100 1000 0010 0010	0x02F24822

Convert the following program from machine language into MIPS assembly language.
The numbers on the left are the instruction addresses in memory, and the numbers on the right give the instruction at that address. Then reverse engineer a high-level program that would compile into this assembly language routine and write it. Explain in words what the program does. $a0 is the input, and it initially contains a positive number, n. $v0 is the output.

Address	Instruction	MIPS
0x00400000	0x20080000	addi $t0, $0, 0
0x00400004	0x20090001	addi $t1, $0, 1
0x00400008	0x0089502A	slt $t2, $a0, $t1
Ox0040000C	0x15400003	bne $t2, $0, 0x0003
0x00400010	0x01094020	add $t0, $t0, $t1
0x00400014	0x21290002	addi $t1, $t1, 2
0x00400018	0x08100002	j 0x0100002
0x0040001C	0x01001020	add $v0, $t0, 0
Ox00400020	0x03E00008	jr $ra

Bkpt	Address	Code	Basic	Source
	0x00400000	0x20100032	addi $16,$0,50	28: addi $16, $0, 50#A = 50;
	0x00400004	0x20110064	addi $17,$0,100	29: addi $17, $0, 100#B = 100;
	0x00400008	0x20120096	addi $18,$0,150	30: addi $18, $0, 150#C = 150;
	0x0040000c	0x02116820	add $13,$16,$17	31: add $13, $16, $17#Temp = A + B;
	0x00400010	0x01b29022	sub $18,$13,$18	32: sub $18, $13, $18#C = A + B - C;
	0x00400014	0x20102710	addi $16,$0,1...	48: addi $16, $0, 10000# G = 10000;
	0x00400018	0x200c2ee0	addi $12,$0,1...	49: addi $12, $0, 12000# C = 12000;
	0x0040001c	0x200d162e	addi $13,$0,5678	50: addi $13, $0, 5678# D = 5678;
	0x00400020	0x018d0822	sub $1,$12,$13	51: sub $1, $12, $13# Temp = C - D;
	0x00400024	0x02018820	add $17,$16,$1	52: add $17, $16, $1# H = G + (C - D);
	0x00400028	0x200c07d0	addi $12,$0,2000	57: addi $12, $0, 2000# C = 2000;
	0x0040002c	0x200d04e2	addi $13,$0,1250	58: addi $13, $0, 1250# D = 1250;
	0x00400030	0x200e1388	addi $14,$0,5000	59: addi $14, $0, 5000# E = 5000;
	0x00400034	0x200f0dac	addi $15,$0,3500	60: addi $15, $0, 3500# F = 3500;
	0x00400038	0x018d1022	sub $2,$12,$13	61: sub $2, $12, $13# Temp = C - D;
	0x0040003c	0x01cf1822	sub $3,$14,$15	62: sub $3, $14, $15# Temp = E - F;
	0x00400040	0x00439020	add $18,$2,$3	63: add $18, $2, $3# I = (C - D) + (E - F);
	0x00400044	0x200a0019	addi $10,$0,25	68: addi $10, $0, 25# A = 25;
	0x00400048	0x200b0032	addi $11,$0,50	69: addi $11, $0, 50# B = 50;
	0x0040004c	0x000b2027	nor $4,$0,$11	70: nor $4, $0, $11# invert bits, NOT operation, ~B;
	0x00400050	0x20850001	addi $5,$4,1	71: addi $5, $4, 1# Temp = ~B + 1;
	0x00400054	0x01458022	sub $16,$10,$5	72: sub $16, $10, $5# G = A - (~B + 1);
	0x00400058	0x200d03e8	addi $13,$0,1000	77: addi $13, $0, 1000# C = 1000;
	0x0040005c	0x200e02ee	addi $14,$0,750	78: addi $14, $0, 750# D = 750;
	0x00400060	0x000e3027	nor $6,$0,$14	79: nor $6, $0, $14# invert bits, NOT operation, ~D;
	0x00400064	0x20c70001	addi $7,$6,1	80: addi $7, $6, 1# Temp = ~D + 1;
	0x00400068	0x00ed7820	add $15,$7,$13	81: add $15, $7, $13# F = C + (~D + 1);
	0x0040006c	0x200a4e20	addi $10,$0,2...	86: addi $10, $0, 20000# A = 20000;
	0x00400070	0x200b7530	addi $11,$0,3...	87: addi $11, $0, 30000# B = 30000;
	0x00400074	0x014b702a	slt $14,$10,$11	88: slt $14, $10, $11# set if less than, compares if one value is less than the other, set to register;
	0x00400078	0x200b0f0f	addi $11,$0,3855	93: addi $11, $0, 0x0F0F# B = 0x0F0F;
	0x0040007c	0x200c0fff	addi $12,$0,4095	94: addi $12, $0, 0x0FFF# C = 0x0FFF;
	0x00400080	0x000b4027	nor $8,$0,$11	95: nor $8, $0, $11# invert bits, NOT operation, ~B;
	0x00400084	0x000c4827	nor $9,$0,$12	96: nor $9, $0, $12# invert bits, NOT operation, ~C;
	0x00400088	0x01620824	and $1,$11,$2	97: and $1, $11, $2# Temp = B & ~C;
	0x0040008c	0x010c1024	and $2,$8,$12	98: and $2, $8, $12# Temp = (~B & C;
	0x00400090	0x00226825	or $13,$1,$2	99: or $13, $1, $2# D = (B & ~C) | (~B & C);

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000