```
.MODEL SMALL ;This directive sets the memory model to small, which means that the code and
data are within the same 64 KB segment. .STACK 100h ;This directive reserves 256 bytes
(100h in hexadecimal) for the stack. .DATA ;This section declares data variables that will be
used in the program. Xsign db ? ;This declares a byte variable Xsign to store the sign bit of the
first number. Ysign DB ? Xexp DB ? ;This declares a byte variable Xexp to store the exponent of
the first number. Yexp DB ? Xsignificand DW ? ;This declares a word variable Xsignificand to
store the significand (mantissa) of the first number Ysignificand DW ? SUMsign DB ? ;This
declares a byte variable SUMsign to store the sign bit of the sum. SUMexp DB ?
SUMsignificand DW ? ExpDiff DB ? ;This declares a byte variable ExpDiff to store the difference
in exponents between the two numbers. .CODE MAIN PROC FAR MOV AX,03700H ;This
instruction moves the hexadecimal value 03700H (14,848 in decimal) into the AX register. MOV
BX,02C00H ;This instruction moves the hexadecimal value 02C00H (11,520 in decimal) into the
BX register PUSH AX ;This instruction pushes the value of AX onto the stack PUSH BX ph0:
;This is a label marking the start of a code block called phase 0 MOV CX,AX AND AX,8000H
;This instruction performs a bitwise AND operation between the value in AX and 8000H, which
masks all bits except the most significant bit (sign bit). ROL AX,01H ;This instruction rotates the
bits in AX to the left by 1 position MOV Xsign,AL ;This instruction moves the lower 8 bits of AX
(which now contain the sign bit) into the variable Xsign MOV AX,CX AND AX,7C00H ;This
instruction performs a bitwise AND operation between the value in AX and 7C00H, which masks
all bits except the exponent bits ROL AX,06H ; MOV Xexp,AL ;This instruction moves the lower
8 bits of AX (which now contain the exponent bits) MOV AX,CX AND AX,03FFH ;This instruction
performs a bitwise AND operation between the value in AX and 03FFH, which masks all bits
except the significand bits ADD AX,0400H ;This instruction adds 0400H (1024 in decimal) to the
value in AX. This is done to account for the implied leading bit in the significand MOV
Xsignificand,AX MOV AX,BX AND AX,8000H ROL AX,01H MOV Ysign,AL MOV AX,BX AND
AX,7C00H ROL AX,06H MOV Yexp,AL MOV AX,BX AND AX,03FFH ADD AX,0400H MOV
Ysignificand,AX ph1: MOV AL,Xexp SUB AL, Yexp ;This instruction subtracts the value of Yexp
from AL, resulting in the difference in exponents. The result is stored in ExpDiff. MOV ExpDiff,AL
JB swp ;Jumps to the "swp" label if the result of the subtraction (ExpDiff) is less than zero
(indicated by the carry flag being set). JMP ph2 swp: MOV CL,Xexp MOV BL,Yexp XCHG
CL,BL ;Swaps the values of CL and BL. MOV Xexp,CL MOV Yexp,BL SUB CL,BL ;Subtracts the
value of BL from CL, storing the result in CL. This calculates the absolute difference between
the exponents and is stored in ExpDiff. MOV ExpDiff,CL ;Moves the result of the subtraction into
the ExpDiff variable. MOV CL,Xsign MOV BL,Ysign XCHG CL,BL MOV Xsign,CL MOV Ysign,BL
MOV CX,Xsignificand MOV BX,Ysignificand XCHG CX,BX MOV Xsignificand,CX MOV
Ysignificand,BX ph2: MOV AL,Xexp MOV SUMexp,AL MOV AL,Xsign MOV SUMsign,AL MOV
AL,Ysign JZ no2C NEG Ysignificand no2C: SHL Xsignificand,03H SHL Ysignificand,03H ph3:
MOV CL,ExpDiff SAR Ysignificand,CL ph4: MOV AX,Xsignificand MOV BX,Ysignificand ADD
AX,BX MOV SUMsignificand,AX AND AH,80H JZ nxt NEG SUMsignificand MOV AL,Ysign Mov
SUMsign,AL nxt: MOV AX,SUMsignificand AND AH,40H JZ ph5 SHR SUMsignificand,01H ADD
SUMexp,01H JZ ph6 ph5: MOV AX,SUMsignificand TEST AH,20H JZ lp JMP ph6 lp: SHL
AX,01H SHL SUMsignificand,01H SUB SUMexp,01H TEST AH,20H JZ lp ph6: MOV
AX,SUMsignificand TEST AX,0004H JZ ph8 TEST AX,0003H JZ ph7 ADD AX,0008H JMP ph8
ph7: TEST AX,0008H JZ ph8 ADD AX,0008H ph8: MOV SUMsignificand,AX MOV CL,SUMsign
SUB CH,CH ROR CX,01H MOV BX,CX MOV CH,SUMexp SHL CH,02H ADD bx,cx SHR
SUMsignificand,03H MOV AX,SUMsignificand AND AX,03FFH ADD BX,AX MOV CX,10H prn:
MOV AH,02H MOV DL,30H TEST BX,8000H JZ zro MOV DL,31H zro: INT 21H SHL BX, 0001H
LOOP prn MOV AH, 4CH INT 21H MAIN ENDP END MAIN
```