## Q1. Assemble the codes. (2 points)

Convert the following 68K assembly language instructions to the machine codes.

1) MOVE.W D1, $0000A000 **ANSWER = 33C1 0000A000 [PAGE 220]**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **00** | | **SIZE** | | **DES. REGISTER** | | | **DES.MODE** | | | **SOU.MODE** | | | **SOU.REGISTER** | | |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| **3** | | | | **3** | | | | **C** | | | | **1** | | | |

2) MOVE.B $42A7, (A1)+ **ANSWER = 12F8 42A7 [PAGE 220]**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **00** | | **SIZE** | | **DES. REGISTER** | | | **DES.MODE** | | | **SOU.MODE** | | | **SOU.REGISTER** | | |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| **1** | | | | **2** | | | | **F** | | | | **8** | | | |

3) ADD.L D7, D0  **ANSWER = D087 [PAGE 108]**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **11** | | **01** | | **REGISTER** | | | **OPMODE** | | | **EA.MODE** | | | **EA.REGISTER** | | |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| **D** | | | | **0** | | | | **8** | | | | **7** | | | |

4) MOVEA.L D3, A0  **ANSWER = 2043 [PAGE 223]**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **00** | | **SIZE** | | **DES. REGISTER** | | | **001** | | | **SOU.MODE** | | | **SOU.REGISTER** | | |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| **2** | | | | **0** | | | | **4** | | | | **3** | | | |

## Q2. Floating point numbers. (2 points)

**WORK SHOWN ON NEXT PAGE. PRETTY BAD HANDWRITING**

A.) Convert the following decimal numbers in IEEE single-precision format. Give the result as eight hexadecimal digits.

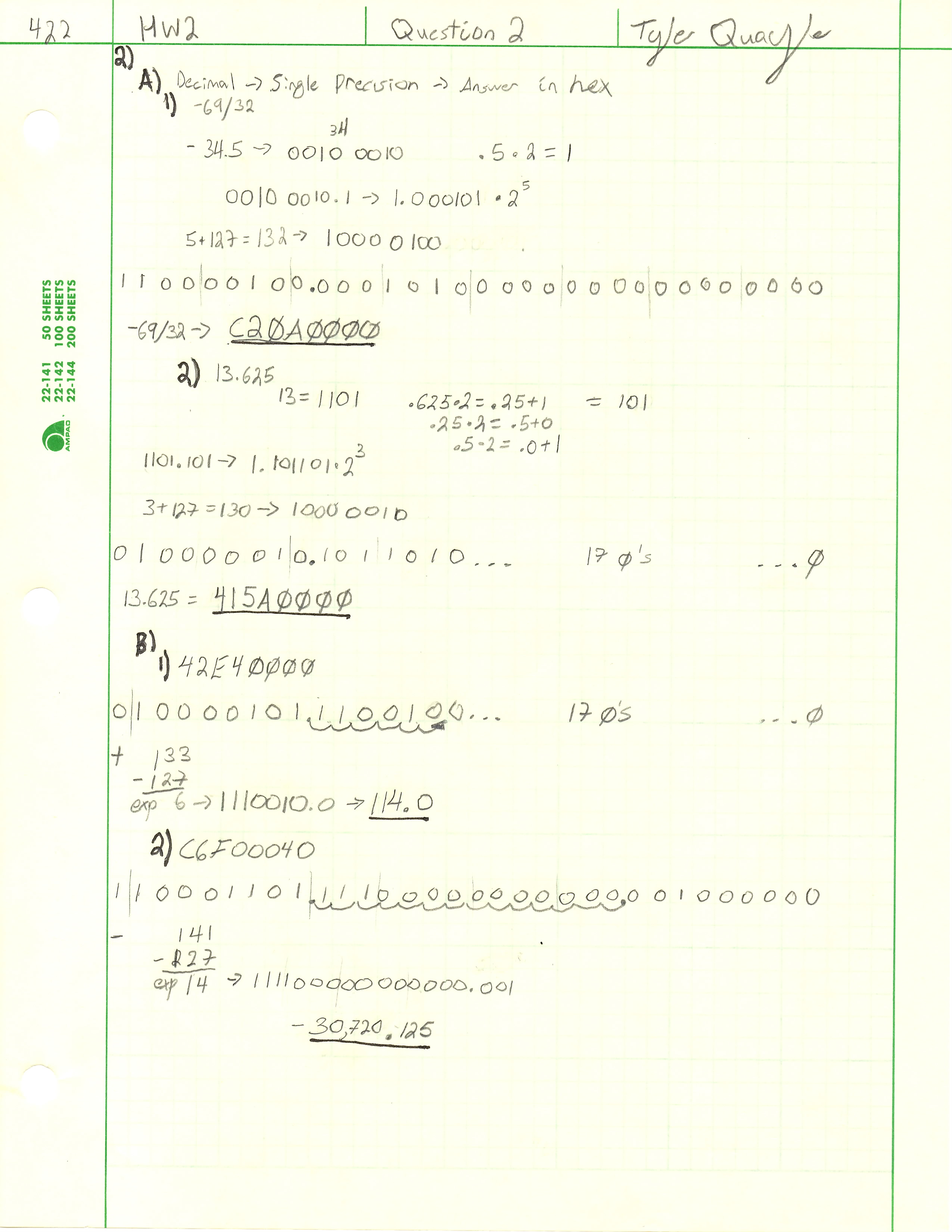
1.) -69/32 (-69 divide by 32) - > **C20A0000**

2.) 13.625 **-> 415A0000**

B.) Convert the following floating IEEE single-precision floating-point numbers from hex to decimal:

1.) 42E48000 -> **114.0**

2.) C6F00040 - > **-30,720.125**



## Q3. Bit Shift (2 points)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **$C1A8E392** | | | | | | | | | | | | | | | | | | | | | | | | **XNZVC** | | | **0** | **0** | **0** | **0** | **0** |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| **C** | | | | **1** | | | | **A** | | | | **8** | | | | **E** | | | | **3** | | | | **9** | | | | **2** | | | |

Assume that D0 contains the value $C1A8E392. For each of the operations below, evaluate the value in D0 and the state of the CCR. Assume that initially XNZVC=00000. (2 points)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| **9** | | | | **2** | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| **4** | | | | **8** | | | |

1. **ASL.B #2, D0**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **$C1A8E348** | | | | | | | | | | | | | | | | | | | | | | | | **XNZVC** | | | **1** | **0** | **0** | **0** | **1** |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| **C** | | | | **1** | | | | **A** | | | | **8** | | | | **E** | | | | **3** | | | | **4** | | | | **8** | | | |

1. **ASL.L #5, D0. Longword, shifts all bits. ASL adds 0’s into LSB. $C1A8E345 >351C6900**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| **C** | | | | **1** | | | | **A** | | | | **8** | | | | **E** | | | | **3** | | | | **4** | | | | **8** | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **$351C6900** | | | | | | | | | | | | | | | | | | | | | | | | **XNZVC** | | | **1** | **0** | **0** | **0** | **1** |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3** | | | | **5** | | | | **1** | | | | **C** | | | | **6** | | | | **9** | | | | **0** | | | | **0** | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **0** | | | | **0** | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **0** | | | | **0** | | | |

1. **LSR.B #4, D0. No change, D0.B is 00, LSR adds 0’s into MSB slot. $351C6900->$351C6900**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **$351C6900** | | | | | | | | | | | | | | | | | | | | | | | | **XNZVC** | | | **0** | **0** | **0** | **0** | **0** |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3** | | | | **5** | | | | **1** | | | | **C** | | | | **6** | | | | **9** | | | | **0** | | | | **0** | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **6** | | | | **9** | | | | **0** | | | | **0** | | | |

1. **ROR.W #2, D0. 351C6900 -> 351C1A40. No 1’s were rotated out. No change to XNZVC**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | | | | **A** | | | | **4** | | | | **0** | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **$351C1A40** | | | | | | | | | | | | | | | | | | | | | | | | **XNZVC** | | | **0** | **0** | **0** | **0** | **0** |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3** | | | | **5** | | | | **1** | | | | **C** | | | | **1** | | | | **A** | | | | **4** | | | | **0** | | | |

## Q4. Pattern Finding and Cumulative program. (5 points)

Write a program in 68K assembly code that satisfy the following specifications:

1. Your program should start at the memory location $1000. Read each **byte data** stored in memory between the addresses $6000 and $8000 and compare it to the (byte) data at address $A000.
2. If it finds the data matching data at address $A000, it stores the **longword address of the data**in memory. Please define a variable called address **Addr1** to save the data. If it fails to find the data within the specified memory range ($6000 and $8000), then put Addr1 = $6000, which is similar to the idea of "error code" or "invalid index" in your C++ code.
3. Then, add a series of **bytes** stored in **256** consecutive memory locations beginning at address Addr1, no matter you find the target data or not.
4. Store the **sum** into memory as a **WORD** variable called **Addsum**. There is a chance that the sum might exceed $FFFF, (exceeding the range of word value), so you will also need to store the carry bit if an overflow occurs. Store the **carry bit as a BYTE** variable called **CarryBit**.
5. Print the **Addr1**, **Addsum** and **CarryBit** in the output window

|  |
| --- |
| \*-----------------------------------------------------------  \*-----------------------------------------------------------  \* Title : Pattern Finding and Cumulative Program  \* Written by : Tyler Quayle  \* Date : April 21, 2016  \* Description: Question 4, in HW2. Find byte in memory, then find SUM of 256 bytes  \*-----------------------------------------------------------  \* FOLLOWING TWO VARIABLES ARE USED TO HARD CODE THE BYTE TO FIND,  \* OTHERWISE WOULD 'SUCCEED', ON FIRST COMPARE SINCE ALL MEMORY DEFAULTS  \* TO 'FF'  NumTo EQU $07 \* Number hardcoded to find  NumLoc EQU $6553 \* Address between $6k-$8k, where variable is 'Hidden'  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  CompA EQU $A000 \* The memory location that is being compared against  StarA EQU $6000 \* Starting Address to search From  EndA EQU $8000 \* End Address to search thru  Addr1 EQU $10F0 \* Defined in HW2, used to track the 'successful' find of matching data  Addsum EQU $10F4 \* Defined in HW2, Used to sum 256 consecutive memory addresses  CarryBit EQU $10F8 \* Defined in HW2, Stores carrybit from ADD Addsum  ORG $1000  START:  MOVEA.L #$6000, A4 \* Assign address $6000 (start) to A4  MOVEA.L #CompA, A3 \* Assign Address $A300 (compare) to A3  MOVE.B #NumTo, NumLoc \* Hardcode number for loop to find  MOVE.B #NumTo, CompA \* Hardcode number that loop to find to A300 (A4/CompA)  MOVE.B (A3),D0 \* Give D0 the byte data in A3  CLR.W Addsum \* Set 4 Bytes to Zero at Addsum address. Otherwise starts summing at FFFF  CLR.W CarryBit \* Set 4 Bytes to Zero, So carry bit is represent correctly.  SEARCHLOOP  ADDA.W #$1,A4 \* Increment the Address of A4 by $1  MOVE.B (A4),D1 \* Give D1 the current Byte A4 is on in the loop  CMP.B D0, D1 \* Compare the byte of data in D0 ($A300) and D1 (A4 Current Byte)  BEQ FOUND \* If D1,D0 Equal, go to find  CMPA.L #EndA, A4 \* Compare current A4 Address to end address  BEQ ERROR \* If EndA hit, jump to error  BRA SEARCHLOOP \* Restart Loop if A) Not found B) not the end  FOUND  MOVE.L A4,Addr1 \* Update Variable Addr1 to Address that was successfully found  BRA CONTINUE \* Jump to CONTINUE  ERROR  MOVE.L #$6000, Addr1 \* 'Error Code' to insert $6000 into Addr1  BRA CONTINUE \* Jump to CONTINUE  CONTINUE  MOVEA.L Addr1, A4 \* Update A4 with the Address of Addr1  CLR.W D1 \* Clear out D1 to Zero, so it can be used to count down properly  CLR.W D3 \* Clear out D3 word, need to Add BYTE but store in WORD.  ADDLOOP  MOVE.B (A4)+, D2 \* Move the Byte value in A4 to D2, than increment A4  ADD.W D2, Addsum \* Add the current Byte in D2 to Addsum  BCS CARRY \* If Addsum exceeds 4 Bytes (Word). BCS  ADD.W #1, D1 \* Add 1 to D1, used to count the 'loops'  CMP.W #255, D1 \* Compare 255 to D1, used to see if D1 is greater than 255  BGT FINISH \* If D1 is greater than 255, finish the program  BRA ADDLOOP \* D1 is still less than 255, loop again  CARRY  MOVE.W #1, CarryBit \* Assign 1 to Carrybit, to represent CCR carrybit  BRA ADDLOOP \* Jump back to ADDLOOP  FINISH  \*-------Display Address----------------------------------------------  LEA AddOut, A1 \* Move AddOut into A1 for display  MOVE.B #14, D0 \* Move (Task)14 into D0 for Trap 15  TRAP #15 \* Display contents of A1  MOVEA.L Addr1, A1 \* Move Address of Addr1 ($6553) into A1  MOVE.W A1, D1 \* Move Address of A1 into D1 for display  MOVE.B #3, D0 \* Move (Task)3[display D1 in decimal] into D0 for Trap 15  TRAP #15 \* Display contents of D1  \*-------Display Sum of Memory-----------------------------------------  LEA SumOut, A1 \* Move SumOut into A1 for display  MOVE.B #14, D0 \* Move (Task)14 into D0 for Trap 15  TRAP #15 \* Display contents of A1  MOVE.W Addsum, D1 \* Move Addsum into D1  MOVE.B #3, D0 \* Move (Task)3[Display D1 in decimal] into D0 for Trap 15  TRAP #15 \* Display contents of D1, in decimal (Task 3)  \*-------Display Carry Bit---------------------------------------------  LEA CarOut, A1 \* Move CarOut into A1 for display  MOVE.B #14, D0 \* Move (Task)14 into D0 for Trap 15  TRAP #15 \* Display contents of A1  MOVE.W CarryBit, D1 \* Move CarryBit into D1  MOVE.B #3, D0 \* Move (Task)3[Display D1 in decimal] into D0 for Trap 15  TRAP #15 \* Display contents of D1, in decimal (Task 3)  SIMHALT ; halt simulator  CR EQU $0D ;ASCII code for Carriage Return  LF EQU $0A ;ASCII code for Line Feed  AddOut DC.B 'Address: ',0  SumOut DC.B LF, CR,'Sum of 256: ',0  CarOut DC.B LF, CR,'Carry Bit: ',0  END START ; last line of source |
|  |
|  |

## Q5. Decode a floating point number. (5 points)

Write a program in 68K assembly code to decipher IEEE 32-bit floating point hexadecimal representation to sign bit, exponent bit, and mantissa bits. Program specifications follow:

1. Your program should start at the memory location $400.
2. The program should print the instruction in the output window to get a user input: "Please enter IEEE 32-bit floating point number in hexadecimal":
3. User Input should be in IEEE 32-bit floating point number Hexadecimal format.
4. After getting the user input, save the number in the memory address right after your program code.
5. Then print out the following information to the output window: Sign bit: ("+" or "-"), Exponent: (in decimal) and Mantissa: (in decimal). For example, if the user input is :C0680000, then the memory should have the data: C0680000 in the address right after your program cod, and then the output window shows:
6. **Sign bit: -   
   Exponent:128**  
   **Mantissa: 13**

|  |
| --- |
| \*-----------------------------------------------------------  \* Title : Decode a Floating Point Number  \* Written by : Tyler Quayle  \* Date : April 22, 2016  \* Description: Given 32-Bit Floating point Hexadecimal, give sign, exponent, mantissa  \*-----------------------------------------------------------  SignBit EQU $580  Exponent EQU $582  Mantissa EQU $584  ORG $400  START: ; first instruction of program  \* Put program code here  LEA InMsg, A1 \* Insert Msg to prompt user into A1 for display  MOVE.B #14, D0 \* Task 14, Trap 15 - Display NULL terminate Sting, no CR/LF  TRAP #15 \* Display Prompt  MOVE.B #2, D0 \* Task 2 for Trap 15, Read in String  TRAP #15 \* Get User input  MOVEA.L #$56A, A5 \* Give A5 Starting Address  MOVE.B #9, D2 \* D2 used for Looping 8 times  CONVERT \* For(D2 = 8, D2 > 0, D2--)  MOVE.B (A1)+,D4 \* Give D4 first Byte of user input, then increment    SUB.B #1, D2 \* For Loop decrement  CMP.B #0, D2 \* Check for end of For-Loop  BEQ MOVEBITS \* Conversion Done, Go To Movebits    CMP.B #$40, D4 \* Compare if D4 contains greater than $40 (A = $41, B = $42)  BGT CONV\_LETTER \* Letter Possibly Found, go to Conv\_Letter to Confirm    CMP.B #$3A, D4 \* Compare D4 if less than $3A (Hex for :) (9 = $39, 8 = $38)  BLT CONV\_NUMBER \* Number possibly Found, go to Conv\_Number to Confirm    CONV\_LETTER  CMP.B #$46, D4 \* Check to see if D4 falls within $41-$46 (A-F)  BGT MISSING \* Not $41-$46 (HEX A-F), Bad/No Data, Jump to Missing  SUB.B #$37, D4 \* Subtract $37, to convert (ASCII->HEX)  MOVE.B D4, (A5)+ \* Give current A5 the D4 hex value and Increment  BRA CONVERT \* Successfully Convert (ASCII->HEX), Branch back to Convert Loop  CONV\_NUMBER  CMP.B #$30, D4 \* Check to see if D4 falls within $30-$39 (0-9)  BLT MISSING \* Not $30-$39 (HEX 0-9), Bad/No Data, Jump to Missing  SUB.B #$30, D4 \* Subtract $30, to convert (ASCII->HEX)  MOVE.B D4, (A5)+ \* Give current A5 the D4 hex value and Increment  BRA CONVERT \* Successfully Convert (ASCII->HEX), Branch back to Convert Loop  MISSING \* Here in case user did not enter 8 numbers, 'fills' rest with 0's  MOVE.B #$00, (A5)+ \* Give Current address Byte value $00, increment A5  BRA CONVERT \* Branch back to Convert Loop  MOVEBITS  MOVEA.L #$56A, A5 \* Reset A5 to starting address  MOVE.B #5, D2 \* Give D2 $5, used in MOVELOOP logic  MOVELOOP \* For(D2 = 5, D2 > 0, D2--)  SUB.B #1, D2 \* For Loop decrement  CMP.B #0, D2 \* Check if conditions have been met  BEQ FINDSIGN \* Bit moving done, D4 Now contains correct 32-Bit Hex  MOVE.B (A5)+, D3 \* Move Current Byte into D3  LSL.L #4, D3 \* Logical Shift Bits 1 Nibble Left (0X->X0)  ADD.B (A5)+,D3 \* Add next Byte into Bit Shifted D3(XX)  LSL.L #8, D4 \* Logical Shift D4 1 Byte (00XX->XX00)  ADD.B D3, D4 \* Add D3 Byte to D4 (XXXX)  BRA MOVELOOP \* Repeat Loop    FINDSIGN  MOVE.L D4, D3 \* Move D4(User Hex Value) into D3 for manipulation  MOVE.B #31, D2 \* Move 31 into D2, used for Following Logic Shift  LSL.L D2, D3 \* Logic-Shift-Left 31, leaving only MSB  MOVE.B #$2D, SignBit \* Default to $2D (Hex for '-')  CMP.B #$1, D3 \* Compare D3 to 1 (check for negative input)  BEQ FINDEXP \* If MSB was negative, continue  MOVE.B #$2B, SignBit \* MSB was 0, change Sign to $2B (Hex for '+')  FINDEXP  MOVE.L D4, D3 \* Move D4(User Hex Value) into D3 for manipulation  BCLR.L #31, D3 \* Clear MSB  MOVE.B #23, D2 \* Move 23 into D2,  LSR.L D2, D3 \* Logic-Shift-Right 23 || Get rid of Mantissa  MOVE.W D3, Exponent \* Give variable Exponent the value in D3  FINDMAN  MOVE.L D4, D3 \* Move D4(User Hex Value) into D3 for manipulation  MOVE.B #9, D2 \* Move 9 into D2, Used for Following Logic Shift  LSL.L D2, D3 \* Logic-Shift-Left 9,  MOVE.B #28, D2 \* Move 28 into D2, Used for following logic shift  LSR.L D2, D3 \* Logic-Shift-Right 28, moving first 4 bits of decimals into D3.B  MOVE.W D3, Mantissa \* Give variable Mantissa value in D3    PRINT  LEA OutSig, A1 \* Load OutSig message into A1 for display  MOVE.B #14, D0 \* Task 14, Trap 15 - Display NULL terminate Sting, no CR/LF  TRAP #15 \* Display OutSig (Sign) message  MOVE.B SignBit, D1 \* Move variable SignBit into D1 for display  MOVE.B #6, D0 \* Task 6, Trap 15 - Display Single Char in D1 (Hex code for +/-)  TRAP #15 \* Display char in D1, Hex value for either '+' or '-'    LEA OutExp, A1 \* Load OutExp message into A1 for display  MOVE.B #14, D0 \* Task 14, Trap 15 - Display NULL terminate Sting, no CR/LF  TRAP #15 \* Display OutExp (Exponent) message  MOVE.W Exponent, D1 \* Move variable Exponent into D1 for display  MOVE.B #3, D0 \* Task 3, Trap 15 - Display Value in D1 as Decimal  TRAP #15 \* Display variable Exponent    LEA OutMan, A1 \* Load OutMan message into A1 for display  MOVE.B #14, D0 \* Task 14, Trap 15 - Display NULL terminate Sting, no CR/LF  TRAP #15 \* Display OutMan  MOVE.W Mantissa, D1 \* Move variable Mantissa into D1 for display  MOVE.B #3, D0 \* Task 3, Trap 15 - Display Value in D1 as Decimal  TRAP #15 \* Display variable mantissa    SIMHALT ; halt simulator  \*---------OutPut Values--------------------------------------------------------  CR EQU $0D ;ASCII code for Carriage Return  LF EQU $0A ;ASCII code for Line Feed  InMsg DC.B CR,'Please enter IEEE 32-bit floating point number in hexadecimal: ' ,0  OutSig DC.B CR, LF,'Sign Bit: ',0  OutExp DC.B CR, LF,'Exponent: ',0  OutMan DC.B CR, LF,'Mantissa: ',0  \* Put variables and constants here  END START ; last line of source |
|  |
|  |