Homework Problem Set #2

**Q1. Convert the following 68K assembly language instructions to the machine codes. (2 points)**

**You have to show your work, otherwise you will get zero**.

1) MOVE.W        D1,  $0000A000

Initial codes from Manual

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 0 1 1 | 0 0 1 | 0 0 0 | 1 1 1 | 0 0 1 |

0011 0010 0011 1001

Hex code: 3239 0000A000

Note: this answer does not agree with the actual assembly when I tested it, but I checked through the Motorola instructions and do not see how the assembler received it answer of 33C1 0000A000.

2) MOVE.B        $42A7, (A1)+

Initial codes from Manual

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 0 0 1 | 0 0 1 | 0 1 1 | 1 1 1 | 0 0 0 |

0001 0010 1111 1000

Hex code: 12F8 42A7

3) ADD.L            D7, D0

Initial codes from Manual

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 1 0 1 | 1 1 1 | 0 1 0 | 0 0 0 | 1 1 1 |

1101 1110 1000 0111

Hex code: DE87

4) MOVEA.L       D3, A0

Initial codes from Manual

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 0 1 0 | 0 0 0 | 0 0 1 | 0 0 0 | 0 1 1 |

0010 0000 0100 0011

Hex code: 2043

**Q2. For each of the operations below, evaluate the value in D0 and the state of the CCR after completing the operation. XNZVC=00000 and D0=$C1A8E392 at the beginning of each operation. (4 points)**

1. ASL.B #2, D0

Operate only on the first byte (2 digits)

The most significant bit changed from a 1 to a 0 at least once, so the V flag is set.

XNZVC=00010

1. ASL.L #5, D0

Operate on entire number

The most significant bit changed from a 1 to a 0 at least once, so the V flag is set.

XNZVC=00010

1. LSR.B #4, D0

Operate on first byte

The last shifted bit (0) is copied to Z and X, V is 0, N and Z are 0

XNZVC=00000

1. ROR.W #2, D0

Operate on first word

The last rotated bit (1) is copied to C, X is not affected, V is 0, Z is 0, N flag is set

XNZVC=01001

**Q3. Pattern Finding and Cumulative program. (5 points)**

**------------------------------------------------Full Code---------------------------------------------------**

\*-----------------------------------------------------------

\* Title :HW2Q3\_jlandron

\* Written by :Josh Landron

\* Date :27Apr2019

\* Description:source code answer to question 3

\*-----------------------------------------------------------

\* Put variables and constants here

\*define LONGWORD variable named Addr1, assign memory location to Addr1

\*if value is found in the range checked, if not found Addr1 should be $6000

Addr1 DS.L 1

\*define WORD variable called Addsum, add all of the next 512 bytes stored

\*in memory beggining at Addr1, save final sum to Addsum

Addsum DS.W 1

\*define BYTE called CarryBit to store the C if it is ever flagged during

\*the summation

CarryBit DS.B 1

\*-----START OF PROGRAM------\*

ORG $1000

START: ; first instruction of program

\*--------------------------------------------------

\* Initial Setup

\*--------------------------------------------------

LEA START\_MSG, A1

MOVE.B #14, D0

TRAP #15

\*take one byte hex value from user,

JSR READ\_CHAR \*Read first digit

JSR AtoH

LSL.B #4,D6

MOVE.B D6,D2 \*save first digit

JSR READ\_CHAR \*read second digit

JSR AtoH

ADD.B D6, D2 \*Add the two HEX digits to get result

MOVE.B D2, $A000 \*save to $A000

CLR.L D2 \*Clear used registers

CLR.L D6

LEA BLANK\_MSG,A1 \*Move output to new line

MOVE.B #14,D0

TRAP #15

\*-------------------------------------------------

\*Loop through memory

\*-------------------------------------------------

\*Read each byte data stored in memory betweeen $6000 and $8000,

\*and compare it to byte stored from user at $A000

MOVEA.L #$6000, A4 \*STARTING LOCATION

MOVEA.L #$8000, A5 \*ENDING LOCATION

MOVE.L #$6000, (Addr1) \*initialize valueS

MOVE.L #$0, (Carrybit)

MOVE.L #$0, (Addsum)

MOVE.B #$2E,$7120 \*Test number

LOOP\_MEM

CMPA.L A4, A5 \*start of loop, break if the addresses are the same

BEQ START\_SUM

MOVE.B (A4),D2

CMP.B ($A000),D2

BEQ SET\_ADDR1

BRA CONTINUE\_MEM

SET\_ADDR1

MOVE.L A4, Addr1

CONTINUE\_MEM

MOVE.B (A4)+,D2 \*Increcemnt A4

BRA LOOP\_MEM

\*----------------------------------------------------

\*Summation loop

\*----------------------------------------------------

\*Set up A4,A5,D2, and D3 for use in summation

START\_SUM

CLR.L D2

CLR.L D3

MOVE.L (Addr1),D4

MOVEA.L D4,A4

ADDI.L #512,D4

MOVEA.L D4,A5

MOVE.B (A4)+,D2 \*Initialize a value into D2, this register will hold values to be added

MOVE.B (A4)+,D3 \*Initialize a value into d3, the sum destination

\*Start summation loop

LOOP\_SUM

CMPA.L A4,A5

BEQ PUSH\_SUM

ADD.W D2,D3

BCS SET\_CARRYBIT \*branch if carry set

BRA CONTINUE\_SUM \*branch past carry setting

SET\_CARRYBIT

MOVE.B #$01,(CarryBit)

CONTINUE\_SUM

MOVE.B (A4)+,D2

BRA LOOP\_SUM

PUSH\_SUM

MOVE.W D3,Addsum

BRA END\_PRINT

\*---------------------------------------------------

\*Ending messages

\*---------------------------------------------------

END\_PRINT

LEA ADDR1\_MSG,A1

MOVE.B #14,D0

TRAP #15

MOVE.L (Addr1),D2

JSR PRINT\_WORD

LEA BLANK\_MSG,A1

MOVE.B #14,D0

TRAP #15

LEA SUM\_MSG,A1

MOVE.B #14,D0

TRAP #15

CLR.L D2

MOVE.W (Addsum),D2

JSR PRINT\_WORD

LEA BLANK\_MSG,A1

MOVE.B #14,D0

TRAP #15

LEA CARRY\_MSG,A1

MOVE.B #14,D0

TRAP #15

CLR.L D2

MOVE.B (Carrybit),D2

JSR PRINT\_BYTE

SIMHALT ; halt simulator

\*--------------------------------------------

\*Subroutines

\*-------------------------------------------

\*----Printing subroutine----

READ\_CHAR

MOVE.B #5, D0

TRAP #15

MOVE.B D1, D6 \*Store char in D6

RTS

\*PARAMETER SHOULD BE STORED IN D6

AtoH

MOVE.B #0, D7

CMP.B #$39, D6 \*30-39 ARE NUMBERS, GREATER IS A LETTER

BGT.B ALPHA

SUBI.B #$30, D6

RTS

ALPHA

SUBI.B #$37, D6

RTS

\*----reading subroutine----

PRINT\_CHAR

MOVE.B #6, D0

MOVE.B D6, D1

TRAP #15

RTS

HtoA

CMPI.B #9, D6

BGT ABCD

NUM

ADDI.B #$30, D6

RTS

ABCD

ADDI.B #$37, D6

RTS

\*-----------------------------------------

\*Printing subroutines for HEX to ASCII

\*----------------------------------------

PRINT\_WORD \*STORE INITIAL WORD IN D2

MOVE.W D2, D6 \*MOVE ENTIRE WORD TO D6 FOR WORK

LSR.W #8, D6 \*shift right to eliminate first 3 digits

LSR.B #4, D6

JSR HtoA

MOVE.B D6, D1

JSR PRINT\_CHAR

MOVE.W D2, D6 \*max shift is one Byte

LSL.L #8, D6 \*shift left to delete first digit

LSL.L #8, D6

LSL.L #4, D6

LSR.L #8, D6 \*shift right to move desired digit to first 4 bits

LSR.L #8, D6

LSR.L #8, D6

LSR.B #4, D6

JSR HtoA

MOVE.B D6, D1

JSR PRINT\_CHAR

JSR PRINT\_BYTE

RTS

PRINT\_BYTE \*STORE INITIAL BYTE IN D2

MOVE.B D2, D6

LSR.B #4, D6

JSR HtoA

MOVE.B D6, D1

JSR PRINT\_CHAR

MOVE.B D2, D6

LSL.B #4, D6

LSR.B #4, D6

JSR HtoA

MOVE.B D6, D1

JSR PRINT\_CHAR

RTS

\*Strings below here

CR EQU $0D

LF EQU $0A

START\_MSG DC.B 'Welcome to Pattern Finding, Please a 2-digit HEX number',CR,LF,0

BLANK\_MSG DC.B '',CR,LF,0

ADDR1\_MSG DC.B 'Addr1 was set to : ',0

SUM\_MSG DC.B 'Addsum is set to : ',0

CARRY\_MSG DC.B 'Carrybit is : ',0

END START ; last line of source

**---------------------------------End of Code-------------------------------**

Output:



(Test number 2E was loaded to address 7120 in memory to show algorithm working).

**Q4. Decode a floating-point number. (5 points)**

**------------------------------------------------Full Code---------------------------------------------------**

\*-----------------------------------------------------------

\* Title :HW2Q4 - Decode 32Bit IEEE floating point number

\* Written by :Joshua Landron

\* Date :30Apr2019

\* Description:Decodes a IEEE floting point number input in HEX

\*-----------------------------------------------------------

\* Put variables and constants here

UserHex DC.L 1 \*VARIABLE TO STORE USER HEX INPUT

ORG $4000

START: ; first instruction of program

\*--------------------------------------------

\*Program Start

\*--------------------------------------------

LEA START\_MSG, A1

MOVE.B #14, D0

TRAP #15

\*Take user input, final hex number will be in D2

JSR READ\_CHAR \*READ FIRST DIGIT

JSR AtoH

MOVE.B D6,D2

LSL.L #4,D2 \*SHIFT D2 LEFT TO MAKE ROOM FOR NEXT DIGIT

MOVE.B #0, D4 \*LOOP COUNTER D4 RESERVED

MOVE.B #6, D5 \*LOOP STOP D5 RESERVED

READ\_LOOP

JSR READ\_CHAR \*LOOP TO READ NEXT 7 DIGITS

JSR AtoH

ADD.B D6,D2

CMP.B D4,D5

BEQ DONE\_READ \*WHEN LOOP IS FINISHED, BRANCH BEFORE SHIFTING

ADDI.B #1,D4

LSL.L #4,D2

BRA READ\_LOOP

DONE\_READ

MOVE.L D2,(UserHex) \*Move user number to variable, leave number in D2

JSR PRINT\_BLANK

\*--------------------------------------------

\*Sign check

\*--------------------------------------------

\*Check sign with and BTST at location 1F(leftmost bit)

BTST #$1F,D2

BEQ PRINT\_POS \*JUMP TO PRINT\_POS IF Z FLAG IS SET

LEA SIGN\_NEG,A1 \*ELSE PRINT NEGATIVE MESSAGE

MOVE.B #14, D0

TRAP #15

BRA START\_EXP

PRINT\_POS

LEA SIGN\_POS,A1

MOVE.B #14, D0

TRAP #15

\*--------------------------------------------

\*Exponant print

\*--------------------------------------------

START\_EXP

LSR.L #$8, D2 \*shift right by 23

LSR.L #$8, D2

LSR.L #$7, D2

LEA EXP\_MSG,A1

MOVE.B #14, D1

TRAP #15

MOVE.B D2, D1

MOVE.B #3, D0

TRAP #15

JSR PRINT\_BLANK

\*--------------------------------------------

\*Mantissa print

\*--------------------------------------------

MOVE.L (UserHex),D2

LSL.L #$1, D2 \*Eliminate sign bit

LSL.L #$8, D2 \*Eliminate exponant

LSR.L #$8, D2 \*Move back to beggining

LSR.L #$1, D2

RIGHT\_ONE

BTST.L #$0,D2

BNE DONE\_SHIFTING

LSR.L #$1,D2

BRA RIGHT\_ONE

DONE\_SHIFTING

LEA MTSA\_MSG,A1

MOVE.B #14, D1

TRAP #15

MOVE.L D2, D1

MOVE.B #3, D0

TRAP #15

SIMHALT ; halt simulator

\*--------------------------------------------

\*Subroutines

\*--------------------------------------------

\*----Reading subroutine----USES D0,D1,D6,D7

READ\_CHAR

MOVE.B #5, D0

TRAP #15

MOVE.B D1, D6 \*Store char in D6

RTS

\*PARAMETER SHOULD BE STORED IN D6

AtoH

MOVE.B #0, D7

CMP.B #$39, D6 \*30-39 ARE NUMBERS, GREATER IS A LETTER

BGT.B ALPHA

SUBI.B #$30, D6

RTS

ALPHA

SUBI.B #$37, D6

RTS

PRINT\_BLANK

LEA BLANK\_MSG,A1 \*Move print to new line

MOVE.B #14, D0

TRAP #15

RTS

\*--------------------------------------------

\*Strings

\*--------------------------------------------

CR EQU $0D

LF EQU $0A

START\_MSG DC.B 'Please enter IEEE 32-bit floating point number in Hexadecimal Format',CR,LF,0

SIGN\_POS DC.B 'Sign bit: +',CR,LF,0

SIGN\_NEG DC.B 'Sign bit: -',CR,LF,0

EXP\_MSG DC.B 'Exponant: ',0

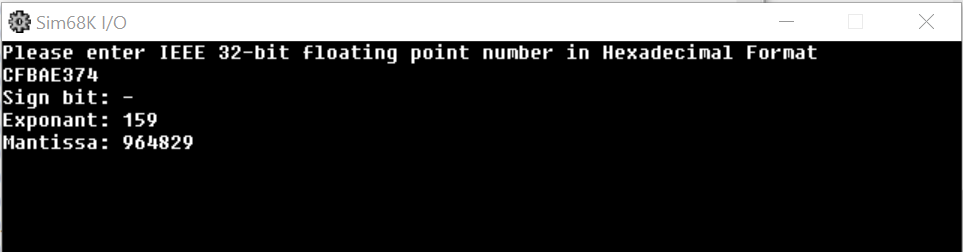
MTSA\_MSG DC.B 'Mantissa: ',0

BLANK\_MSG DC.B '',CR,LF,0

END START ; last line of source

**---------------------------------End of Code-------------------------------**

Output:



(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)

