**Bisrat Asefaw**

**CSS 422 (Hardware)**

**Professor Yang**

**Homework #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

Based on manual and cheat sheet

**Op-code size dest. register dest. Mode source Mode source register**

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

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

1. For MOVE opcode the 15th and 14th bits are always 0, 0 respectively
2. Since the opcode we have has a Word (.W) size bits 13th and 14th are filled with 1, and 1 respectively.
3. Destination Register for, $0000A000 which falls in the longword region fills the bits represented by 11th, 10th, and 9th are 0 0 1 respectively and the destination mode for this value is 1, 1, 1 and fills bits 8th, 7th,and 6th respectively
4. Source mode for D from the manual is 0, 0, 0 and these fills bits that are numbered 5th, 4th 3rd respectively, and the source register is 0, 0, 1 since it is D1(1 represented by 0 0 1), and they filled up the bits numbered by 2nd ,1st , and 0 respectively.
5. The final binary representation is going to be 0011 0011 1100 0001 and its hex representation by grouping 4 bits at a time is 33C1.

**Ans: 33C1 0000A000**

2) MOVE.B        $42A7, (A1)+

**Op-code size dest. register dest. Mode source Mode source register**

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

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

1. For MOVE the Opcode 15th and 14th bits are always 0, 0 respectively
2. Since the given opcode has a size of byte (.B) bits 13th and 12th are filled by 0,and 1 respectively.
3. Destination mode is (A1)+, from the manual its representation is 0, 1, 1 and this fills up bites numbered 8, 7, and 6 respectively, and the destination register represented by bits numbered 11,10, and 9 are filled by 0, 0 and 1 because 1🡪 0 0 1
4. Source mode for the address $42A7 which falls in a **word** zone and fills bits 11th, 10th, 9th are 1, 1, and 1 respectively. The source register for this value is 0,0,0 because it is a word and fills bits 8th 7th 6 respectively
5. Final binary representation is 0001 0010 1111 1000

**Ans: 12F8 42A7**

3) ADD.L            D7, D0

**Op-code dest. register (size .LONG) source Mode source register**

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

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

1. For ADD the opcode binary representation for the bit number 15th, 14th, 13th, and 12th are 1, 1, 0, and 1 respectively.
2. The destination-register bites numbered 11th, 10th, and 9th are filled with 0, 0, 0 respectively. Bite number 8th is filled with 0 because it is adding from data register to data register (Dn🡪Dn).
3. Since the size is LONG, we represent bits 7th and 6th by 1, 0.
4. I represented the bit number 5th,4th,3rd because the source Mode is Data (Dn) and the preceding bits 2nd, 1st, and 0 are filled with 1 1 1 because the source register is D7 (7 binary is 1 1 1)
5. The final binary representation of the instruction is 1101 0000 1000 0111

ANS: **D087**

4) MOVEA.L       D3, A0

**Op-code size dest. register dest. Mode source Mode source register**

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

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

1. I filled the bits numbered 15 th,14th with 0, 0 resp. because the binary representation of the opcode MOVEA is 0, 0
2. The next bits numbered 13 th,12th are filled with binary digit 1, and 0 respectively because we have a MOVEA.L operation with size long (.L).
3. The destination register of this operation is 0, 0, 0 and these fills the bits numbered 11th,10th,9th respectively and the destination mode is clearly an Address (An) and represented by 0,0,1 and fills the bits numbered 8th, 7th, and 6th respectively
4. The source mode of this operation is Data register (D3) and it is represented by 0,0,0 and fills the bits numbered by 5th, 4th, 3rd respectively and its data register number is 3 and represented by 0,1,1 and fills the bits numbered 2nd, 1st, and 0 respectively
5. The final binary representation is 0010 0000 0100 0011

**Ans: 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

* Since the shifting size is a byte (.B) it shifts the value 9 2 (10010010) when shifted two times to the left it becomes 4 8(01001000)

**Ans**:

**D0=C1A8E348**

**XNZVC=00010**

2. ASL.L #5, D0

Long athematic shifting of D0🡪 1100 0001 1010 1000 1110 0011 1001 0010 after shifting 5 times to the left becomes 0011 0101 0001 1100 0111 0010 0100, which is shown below.

**Ans:**

**D0=351C7240**

**XNZVC=00010**

3. LSR.B #4, D0

Shift right 4 times of size .B from D0=$C1A8E392 ==>deals with 9 2 represent by 10010010 after shifting 4 times becomes 0000 1001🡺 hex representation by 09

**Ans:**

**D0=C1A8E309**

**XNZVC=00000**

4. ROR.W #2, D0

Rotate right of, .W length of bits from D0 represented by $C1A8E392 and since the length is Word it rotates the values $E392(01110001110010010) to the right after shifting 1011100011100100 🡺 B8E4

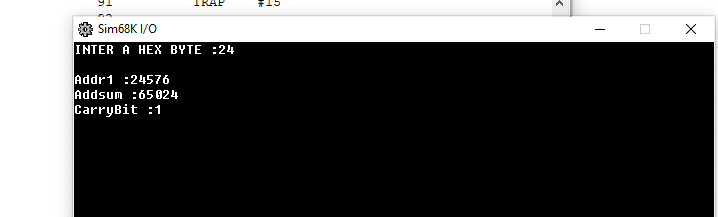
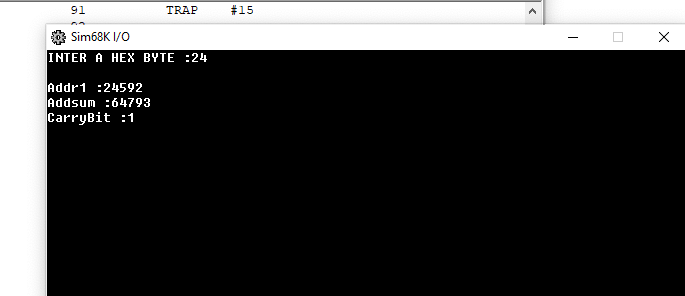
**D0=C1A8B8E4**

**XNZVC=01001**

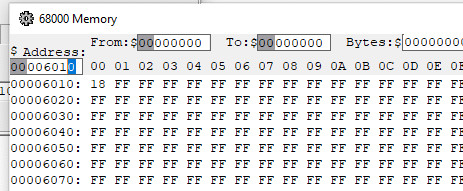
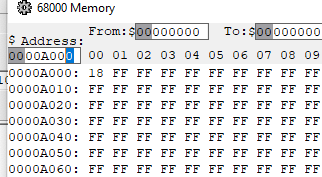
**Q3. Pattern Finding and Cumulative program.**

I tested my program by putting a target at address $6010 and various places between 6000-8000 and see the Addr1 if it matched with the target address. I also changed my target position in different places but between $6000-$8000 and they come up to have expected output as shown in the screen shot below. That is how I tested my program and came up to say it is running perfectly and have a correct output and it clearly shows that my program is working as expected.

Test 1 output target=$6010 Test 2 when there is not target value



Memory location target= $6010 Memory picture of location A000



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

\* Title : HOMEWORK\_2 QUESTION 3

\* Written by : BISRAT ASEFAW

\* Date : 11/22/2020

\* Description: ADDR1,ADDSUM,CARRYBIT

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

Addr1 DS.L 10

Addsum DS.W 10

CarryBit DS.B 10

TARGET EQU $A000

ORG $1000

START: ; first instruction of program

\* Put program code here

LEA InMessage,A1

MOVE.B #14,D0

TRAP #15

MOVE.B #4,D0

TRAP #15

\*MOVE.B #24,$6010 \*TEST CASE 1

MOVE.B D1,$A000

\*CLR.L D1

MOVE.B $A000,D1

MOVEA.L #$6000,A2

CLR D2

MOVE.B TARGET,D2

LOOP CMPA.L #$8000,A2

BGT NOTFOUND

CMP.B (A2)+,D2

BEQ FOUND

BRA LOOP

NOTFOUND MOVE.L #$6000,Addr1

MOVE.L Addr1,A2

BRA ADDALL

FOUND ADD.L #-1,A2

MOVE.L A2,Addr1

BRA ADDALL

ADDALL MOVEA.L A2,A5

ADDA.L #512,A5 \*MOVE.B #0,CarryBit \*FOR TESTING

CLR.B CarryBit \*MOVE.W #0,Addsum \*FOR TESTING

CLR.W Addsum

CLR D1

CLR D5

\*CLR D2 FOR TESTING

ADD CMPA.L A5,A2

BGE DONE

\*ADD #1,D2 \*FOR TESTING

MOVE.B (A2)+,D1

ADD.W D1,Addsum

BCS CARRYDETECTED

BRA ADD

CARRYDETECTED MOVE.B #1,CarryBit

BRA ADD

DONE LEA ADDRESSPRINT,A1

MOVE.B #14,D0

TRAP #15

CLR D1

MOVE.L Addr1,D1

MOVE.B #3,D0

TRAP #15

CLR D1

LEA ADD\_SUM,A1

MOVE.B #14,D0

TRAP #15

\*MOVE.W D5,Addsum TESTING

CLR.W D1

MOVE.W Addsum,D1

MOVE.B #3,D0

TRAP #15

LEA Carr\_yBit,A1

MOVE.B #14,D0

TRAP #15

CLR.L D1

MOVE.B (CarryBit),D1

MOVE #3,D0

TRAP #15

SIMHALT ; halt simulator

\* Put variables and constants here

CR EQU $0D

LF EQU $0A

InMessage DC.B 'INTER A HEX BYTE :',0

ADDRESSPRINT DC.B CR,LF,'Addr1 :',0

ADD\_SUM DC.B CR,LF,'Addsum :',0

Carr\_yBit DC.B CR,LF,'CarryBit :',0

END START ; last line of source

**Q4. Decode a floating-point number.**

I used different inputs to test my program some of them by calculating different floating-point numbers ( -3.34375= $C0560000) using hand and test them on the program to see if they have same answers. I provided the screen shot of the test case with $**C0560000** on the end of this section. I use the provided IEEE-32 floating point number and the program runs and produce expected output, which as the values given on this assignment portal. That’s how I tested my program and it clearly gives expected output.

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

\* Title : CHANGE FROM IEEE-32 TO SIGN, EXPONENT,MANTISA

\* Written by : BISRAT ASEFAW

\* Date : 10/23/2020

\* Description: PROGRAM TO CHANGE FROM HEX-IEEE TO SIGN, EXPONENT,MANTISA

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

USERINPUT EQU $5000

USERINPUTDATA EQU $5010

UERINPUTINHEX EQU $5020

ORG $4000

START: ; first instruction of program

\* Put program code here

LEA MESSAGE,A1

MOVE.L #14,D0

TRAP #15

MOVE.L #2,D0

TRAP #15

SUBI.L #2,D1 \*SUBTRACTS 2 TO THE LENGTH OF CHARACTER INPUT B/C IT SHIFTS TWO CHAR

\* CAUSING TWO CHARACTER TO BE SHEFTED OUT OR BECOMING OVERFLOW

MOVE.L A1,USERINPUT \*SAVING USER INPUT ADDRESS

MOVE.L (A1),USERINPUTDATA \*SAVING USER INPUT DATA

CLR.L D2 \*CLEARING DATA REGISTERS D2,D3 FOR STORING

CLR.L D3

CHANGE MOVE.B (A1)+,D2 \*LOOPING THROUGH USER INPUT TO CONVERT TO HEX VALUE

CMP.B #0,D1

BEQ DONE \* WHEN WE FINISH CHANGING AND SAVE THE INPUT CHARACTER ON D3

CMP.B #$40,D2 \* WE BRACH TO DONE TO PERFORM SPLITING OF SIGN,EXPONENT, MANTISSA

BGE WORD\_CHANGE \*BRANCH FOR A CHARACTER A-F

BLT NUM\_CHANGE \*BRANCH FOR A NUMBER 0-9

WORD\_CHANGE SUB.B #$37,D2 \* CHARACTER CONVERTER BY SUBTRACTING 37 AND SAVING IT IN D2

LSL.B #4, D2 \* LOGIC SHIFT LEFT TO CREATE A SPACE FOR POSITIONING INPUTS

MOVE.B D2, D3 \* MOVE IT TO D3 TO ACCUMULATE CHANGED INPUTS

LSL.L #4, D3 \* LOGIC SHIFT LEFT TO CREATE A SPACE FOR OTHER INPUTS

SUB.B #1,D1 \* SUBTACT 1 FROM THE NUMBER OF CHARACTERS INPUT

BRA CHANGE \*TAKES BACK TO THE BEGINIGN TO REPEAT CHANGING CHAT BY CHAR

NUM\_CHANGE SUB.B #$30,D2 \*CHANGES THE INPUT CHARACTER FOR NUMBER AND DO THE NECESSARY SHIFTING

LSL.B #4, D2

MOVE.B D2, D3

LSL.L #4, D3

SUB.B #1,D1

BRA CHANGE \*TAKES BACK TO THE BEGINIGN TO REPEAT CHANGING CHAT BY CHAR

DONE MOVE.L D3,D1 \* START CHANGING THE INPUT

MOVE.L D3,UERINPUTINHEX

\*CARRY-BIT

LEA SIGN,A1 \*DISPLAY THE SIGNBIT MESSAGE

MOVE.B #14,D0

TRAP #15

LSL.L #1,D1

BCS NEGATIVENUM \*BRANCH FOR NEGATIVE NUMBER

LEA POSNUM,A1 \*DISPLAY POSITIVE SIGN

MOVE.B #14,D0

TRAP #15

BRA EXPONENTNUM \*BRANCH FOR EXPONENT

NEGATIVENUM LEA NEGNUM,A1 \*DISPLAY NEGATIVE SIGN

MOVE.B #14,D0

TRAP #15

EXPONENTNUM LEA EXPONENT,A1 \*DISPLAY THE EXPONENT MESSAGE

MOVE.B #14,D0

TRAP #15

ROL.L #8,D1 \*PERFORM SHIFTING FOR EXPONENT

MOVE.L D1,D3

CLR.L D1

MOVE.B D3,D1

MOVE.B #3,D0

TRAP #15

LEA MANTISA,A1 \*DISPLAY THE MANTISSA MESSAGE

MOVE.B #14,D0

TRAP #15

LSR.L #8,D3 \*PERFORM SHIFTING FOR MANTISSA

LSR.L #1,D3

\*PUSH THE ZEROS OUT

PUSHZEROS BTST #$0,D3

BNE CLEARED

LSR.L #1,D3

BRA PUSHZEROS

CLEARED CLR.L D1

MOVE.L D3,D1

MOVE.B #3,D0 \*DISPLAY MANTISSA VALUE

TRAP #15

SIMHALT ; halt simulator

\* Put variables and constants here

CR EQU $0D

LF EQU $0A

MESSAGE DC.B 'Please enter IEEE 32-bit floating point number in hexadecimal: ',0

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

EXPONENT DC.B CR,LF,'EXPONENET: ',0

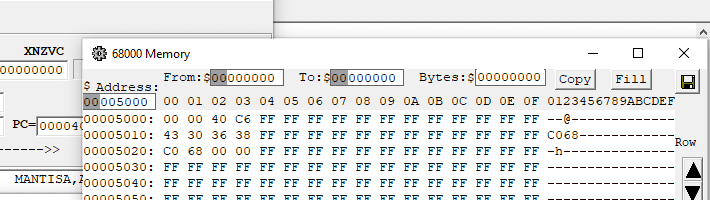
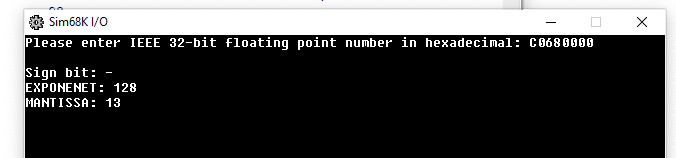
MANTISA DC.B CR,LF,'MANTISSA: ',0

POSNUM DC.B '+',0

NEGNUM DC.B '-',0

END START ; last line of source

FIG. testing by the provided value C0680000 Memory figure



I use a number decimal **-3.34375** = $ C0560000 to test my program by following the steps:

1. First, I converted the decimal -3.34375 to floating point IEEE-32 by hand.
2. I run the result to check if they have the same result.

Figure result of C0560000 Memory picture of C0560000

