Q1)

0-ADDRESS

PUSH A; PUSH A ONTO THE STACK

PUSH B; PUSH B ONTO THE STACK

PUSH C; PUSH C ONTO THE STACK

MUL; MULTIPLYT B AND C

ADD; ADD A TO CURRENT RESULT

PUSH D; PUSH D ONTO THE STACK

PUSH E; PUSH E ONTO THE STACK

PUSH F; PUSH F ONTO THE STACK

MUL; MULTIPYT E AND F

SUB; SUBTRACT E AND F FROM D

DIV; DIVIDE THE RESULTS

POP X; RETURN X

1-ADDRESS

LOAD E; LOAD THE VALUE OF E

MULT F; MULTIPLY E AND F

STORE Y; STORE THE MULTIPLICATIVE IN Y

LOAD D; LOAD CALUE D

SUB Y; SUBTRACT COMPUTED Y BY D

STORE Y; STORE (D-E*F)

LOAD B; LOAD VALUE B

MUL C; MULTIPLY B*C

ADD A; ADD TO B*C

DIV Y; DIVIDE A+B*C/Y

STORE X; TORE COMPUTED VALUES IN X

2- ADDRESS

MOVE R1, E; MOVE THE VALUE OF E INTO FRIST REGISTER

MULT R1, F; MULTIPLY THE VALUE IN THE REGISTER BY F

MOVE R2, D; MOVE VALUE D IN ANOTHER REGISTER 2

SUB R2, R1; BUBTRACT R1 FROM R2 AND STORE IN R2

MOVE R1, B; MOVE B INTO REG 1

MUL R1, C; MOUTIPLY VALUE B*C AND STORE IN R1

ADD R1, A; ADD A TO R1

DIV R1, R2; DIVIDE R1 BY R2 AND STORE VALUE IN R1

MOVE X, R1; MOVE VALUES OF R1 TO X

3- ADDREESS

MULT R1, E, F; MULTIPLY E*F AND STORE IN R1

SUB R1, D, R1; SUBTRACT R1 FROM D AND STORE IN R1

MULT R2, B, C; MULTIPLY B*C AND STORE IN R2

ADD R2, A, R2; ADD A TO R2 AND STORE IN R2

DIV X, R1, R2; DIVIDE R1 AND R2 AND STORE IN X

Q2)

20 – 40

30 – 50

40 – 60

50 – 70

A) 20

B) 40

C) 60

D) 30

E) 50

F) 70

Q3)

A)

AS11M 01.05 Thu Apr 01, 2021 23:36 reversingArray.lst

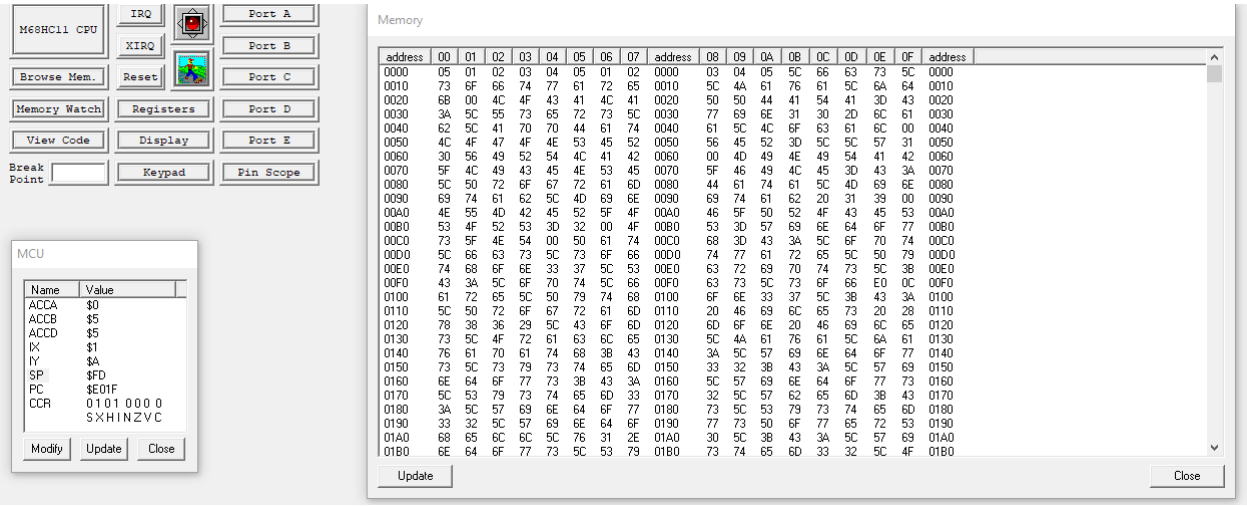
```
0001 0000                       ORG     $0000;   start data at 0000
0002 0000 05            N       FCB     5;      array size
0003 0001 01 02 03 04 05        ARR     FCB     1,2,3,4,5;     array values
0004 0006 00            NEW     FCB;
0005 e000                       ORG     $E000;  continue with program at E000
0006 e000 ce 00 01              LDX     #ARR;   load the address of the arrayS END
0007 e003 18 ce 00 06           LDY     #NEW;
0008 e007 96 00                 LDAA    N;      load the size of the array
0009
0010 e009 bd e0 0d              JSR     LOAD;   call the subroutine
0011 e00c 01                    NOP;            this is where your program ends
0012                    ;               (you may need to set a
0013                    ;               breakpoint here)
```

```
0014                    ;
0015 e00d e6 00         LOAD            LDAB    0,X;
0016 e00f 18 e7 00                      STAB    0,Y;
0017 e012 18 08                         INY;
0018 e014 08                            INX;
0019 e015 4a                            DECA;
0020 e016 26 f5                         BNE     LOAD;
0021 e018 18 ce 00 0a                   LDY     #NEW+4;
0022 e01c ce 00 01                      LDX     #ARR;
0023 e01f 96 00                         LDAA    N;
0024 e021 bd e0 24                      JSR     REVERSE1;
0025
0026 e024 18 e6 00      REVERSE1        LDAB    0,Y;
0027 e027 e7 00                         STAB    0,X;
0028 e029 08                            INX;
0029 e02a 18 09                         DEY;
0030 e02c 4a                            DECA;
0031 e02d 26 f5                         BNE     REVERSE1;
0032 e02f 39                            RTS;
0033
```

Number of errors 0

Number of warnings 0

B)

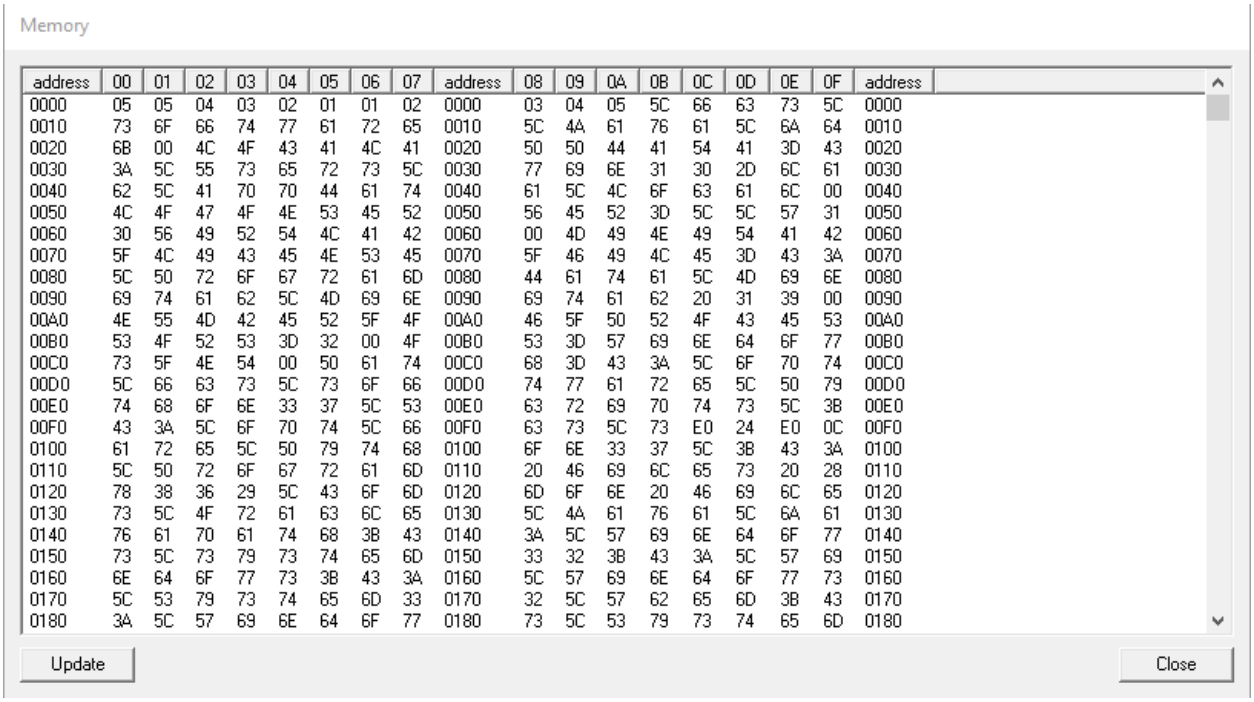


C)

00FE – E0

00FF – 0C

D)



E)

ARR	NEW	ARR	NEW	RETURN

5	5	1	5	
4	4	2	4	
3	3	3	3	
2	2	4	2	
1	1	5	1	
*	*	*	*	*