EEAP 282

EXAM #3
SOLUTIONS

November 17, 1997

NAME: _____ CWRUnet ID:_____

IMPORTANT INFORMATION:
Exam is closed book, closed notes. Only the M68000 Programmer's Reference Manual and/or Programming Reference Card are allowed to be used. NOT ALL PROBLEMS COUNT THE SAME.

Problem	Score	Possible
1 Shift & Rotate		15
2 Branches & Loops		10
3 Math Routines		10
4 Subroutine #1		20
5 Subroutine #2		25
6 Subroutine #3		20
TOTAL SCORE		100

Hint:

```
        quotient
      ___________
divisor | dividend
```

Number of people who tookthe exam: 40+78+33=151

Name: _____ CWRUnet ID:_____

1. The following code is executed:

```
LSR.W     #1,D0
ROR.W     #1,D1
MOVEM.L   D0-D1,-(A0)
```

Memory is initially as given below

```
$1100    [ $00 ]    -->  [ $CD ]
$1101    [ $A4 ]    -->  [ $23 ]
$1102    [ $FF ]    -->  [ $2D ]
$1103    [ $A2 ]    -->  [ $1F ]
$1104    [ $FE ]    -->  [ $BF ]
$1105    [ $00 ]    -->  [ $80 ]
$1106    [ $A4 ]    -->  [ $14 ]
$1107    [ $A2 ]    -->  [ $94 ]
$1108    [ $EC ]
```

You may further assume

```
(A0)=$00001108
(D0)=$CD235A3E
(D1)=$BF802928
```

before the above code is executed. What are D0, D1, A0 and the memory contents after the code is executed?

(D0) = _____ ANSWER: $CD232D1F 4 points

(D1) = _____ ANSWER: $BF801494 4 points

(A0) = _____ ANSWER: $00001100 2 points

MEMORY 5 points.

ANSWER:
The first instruction logically shifts D0.W to the right 1 bit to give
(D0)= $CD232D1F (5 points)
The second instruction does a rotate right by 1 bit on D0.W to give
(D1)=$BF801494 (5 points)
The last instruction is a push to memory of 8 bytes which changes the above memory as indicated. (5 points)

Name: _____ CWRUnet ID:_____

2. What is in D0 and D1 after the following program is executed?

```
     MOVE.L    #$0A00FFFF,D0
     MOVEQ.L   #31,D1
NB:  ASL.L     #1,D0
     DBCS      D1,NB
     TRAP      #0
```

(D0) = _____ ANSWER: $ 401FFFE0 5 points

(D1) = _____ ANSWER: $ 0000001B 5 points

ANSWER:
This program shifts D0 to the left until a one is encountered and leaves the character position of where the one was encountered

```
$0A00FFFF     C=0     D1=$0000001F
$1401FFFE     C=0     D1=$0000001E
$2803FFFC     C=0     D1=$0000001D
$5007FFF8     C=0     D1=$0000001C
$A00FFFF0     C=0     D1=$0000001B
$401FFFE0     C=1     D1=$0000001B
Note that the last iteration the value of D1 does not
decrement because the carry bit is set and the program
control falls through to the next instruction.
```

```
CODE
     ORG       $2000
     MOVE.L    #$0A00FFFF,D0
     MOVEQ.L   #31,D1
NB:  ASL.L     #1,D0
     DBCS      D1,NEXTBIT
     TRAP      #0
```

Name: _____ CWRUnet ID:_____

3. What is in D5 after executing the following program fragment ?

```
ORG       $3000
MOVE.W    #$FFB3,D4     ;-77 decimal
MOVE.L    #$109,D5      ;265 decimal
DIVS      D4,D5
SWAP      D5
```

Answer:
The DIVS instruction performs a SIGNED divide of 265 by -77.
The quotient is -3 ($FFFD). The remainder is 34 ($0022).
The result of the DIVS is then (quotient | remainder) = $0022FFFD.
The SWAP instruction switches the upper and lower words giving the result $FFFD0022.

(D5) = _____ ANSWER: $FFFD0022 5 points

Name: _____ CWRUnet ID:_____

4. What are the contents of the 10 words in memory beginning at $4600? The emphasis upon you to understand the algorithm. A simple flow chart or pseudocode would be very useful, especially for partial credit.

```
FIB  EQU     $4600

     ORG     $4000
     MOVE.L  #$11,D0        ;SET THE COUNTER TO 17
     LEA     FIB,A1
     CLR.W   D1             ;D1=0
     MOVEQ.W #1,D2          ;D2=1
     MOVE.W  D1,(A1)+       ;STORE THE FIRST NUMBER
     MOVE.W  D2,(A1)+       ;STORE THE SECOND NUMBER

NXT: JSR     SBR
     MOVE.W  D2,D1          ;RA#1
     MOVE.W  D3,D2
     MOVE.W  D3,(A1)+
     DBF     D0,NXT

     BRA     DONE

SBR: ADD.W   D2,D1
     MOVE.W  D1,D3
     RTS

DONE END
```

Name: _____ CWRUnet ID:_____

ANSWER: This will be a difficult problem to grade.
This program computes the Fibonacci series according to the following pseudocode:

```
     D1=0; D2=1
     push D1 onto stack incrementing A1
     push D2 onto stack incrementing A1

loop:
     D3=D1+D2
     D1<--D2
     D2<--D3
     push D3 onto stack incrementing A1
     {D0<--D0-1; if D0≠-1 then goto loop}

     done
```

15 points for the flow chart or pseudocode.
10 points for the stack contents.

The stack contains:

```
     $4600    [        ]    -->  [ $0000 ]
     $4602    [        ]    -->  [ $0001 ]
     $4604    [        ]    -->  [ $0001 ]
     $4606    [        ]    -->  [ $0002 ]
     $4608    [        ]    -->  [ $0003 ]
     $460A    [        ]    -->  [ $0005 ]
     $460C    [        ]    -->  [ $0008 ]
     $460E    [        ]    -->  [ $000D ]
     $4610    [        ]    -->  [ $0015 ]
     $4612    [        ]    -->  [ $0022 ]
```

Detailed program operation:
Lets start following the program through beginning at $4000.
The MOVE puts $11 (decimal 17) into D0, i.e., (D0.L)=$00000011
The LEA puts the address FIB ($00004600) into A1.
The CLR sets (D1.L)=$00000000
The MOVEQ sets (D2.L)=$00000001
The first MOVE.W pushes $0000 onto the data stack.
The second MOVE.W pushes $0001 onto the data stack and advances A1 to $00004604 .

We next go to the JSR SBR. This pushes RA#1 onto the program stack. SBR computes D1=D1+D2=$0001, puts this result into D3, and returns popping RA#1 from the program stack.

We now move D2=$0001 into D1, copy D3=$0001 into D2, and push D3=$0001 onto the data stack. We then decrement D0 to D0=$00000010 and branch to NXT since D0 is not equal to -1.
D0=$0010
D1=$0001

Name: _____ CWRUnet ID:_____

D2=$0001
D3=$0001

This calls SBR pushing RA#1 onto the stack. SBR computes D1=D1+D2 =$0001+$0001 =$0002, put this result into D3, and returns popping RA#1 from the stack.

We now move D2=$0001 into D1, copy D3=$0002 into D2, and push D3=$0002 onto the data stack. We then decrement D0 to D0=$0000000F and branch to NXT since D0 is not equal to -1.

D0=$000F
D1=$0001
D2=$0002
D3=$0002

This calls SBR pushing RA#1 onto the stack, computes D1=D1+D2 =$0001+$0002 =$0003, puts this result into D3, and returns popping RA#1 from the stack.

We now move D2=$0002 into D1, copy D3=$0003 into D2, and push D3=$0003 onto the data stack. We then decrement D0 to D0=$0000000E and branch to NXT since D0 is not equal to -1.

D0=$000E
D1=$0002
D2=$0003
D3=$0003

This calls SBR pushing RA#1 onto the stack, computes D1=D1+D2 =$0002+$0003 =$0005, puts this result into D3, and returns popping RA#1 from the stack.

We now move D2=$0003 into D1, copy D3=$0005 into D2, and push D3=$0005 onto the data stack. We then decrement D0 to D0=$0000000D and branch to NXT since D0 is not equal to -1.

D0=$000D
D1=$0003
D2=$0005
D3=$0005

etc.

Name: _____ CWRUnet ID:_____

5. A student has decided to use in line coding of data to pass parameters to a subroutine and the stack to return a single word length result. The main program shown below calls the subroutine SUBR. The stack pointer is initially at $8000. Answer the following questions:

(a) What is on the stack when the PC is at the label INST? Explicitly show all stack contents AND addresses.

```
    final SP--> [  D1  ]
                [  D1  ]
                [  D3  ]
                [  D3  ]
                [  RA  ]
original SP-->  [  RA  ]
```

ANSWER: Shown as a word width stack. If you got the order and size right you typically got full credit (5 points)

```
       ORG        $6000
main   MOVE.W     #6,D1
       MOVE.W     #5,D2
       ADD        D1,D2
JSR    SUBR
A      DC.L       4
B      DC.W       2
C      DS.W       1
* Your subroutine should return to the following instruction.
* and pop a word length result off the stack
DOIT   MOVE.W     (SP)+,C
       END        main

SUBR   MOVEM.L    D1/D3,-(SP)
```

```
*(b) Write instructions to put A into D1 and B into D2.
*You are NOT allowed to use the symbols A and B in your code.

ANSWER:  MOVE.L   8(SP),A0
         MOVE.L   (A0)+,D1    ;get A
         MOVE.W   (A0)+,D2    ;get B, A0 pointing to C
* worth 7 points; -4 points if you calculated an address
* and used immediate addressing
```

```
INST   MOVE.L      #1,D3
       MULS        D1,D3          ;answer in D3
```

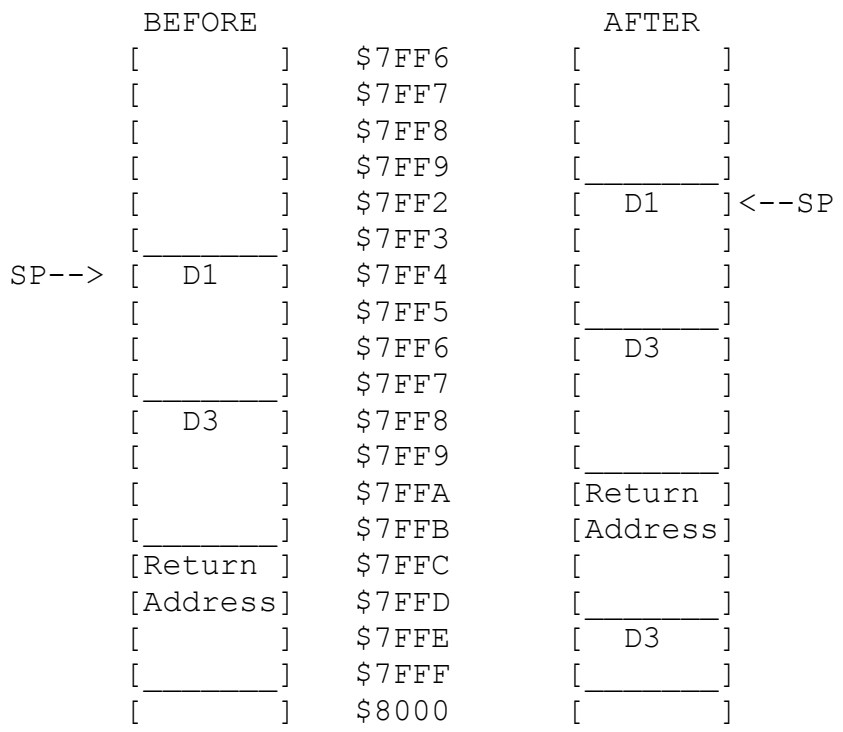
Name: _____ CWRUnet ID:_____

```
*(c) Write instructions to put word length result in D3 onto
stack such that it can be popped off stack after subroutine
return at DOIT.

*This was worth 13 points with -7 points for not making
room for the answer on the stack.  There were a lot of
different answers.  The following is typical:
       MOVE.L      (SP),-2(SP)   ;move D1
       ADDQ.L      #-2,SP
       MOVE.L      6(SP),4(SP)   ;move D3
*   5 points for properly retrieving D1 and D3
       ADDQ.L      #2,A0
       MOVE.L      A0,8(SP)      ;put correct RA in place
*   5 points for putting correct return address in place
       MOVE.W      D3,$C(SP)     ;put answer on stack
*   2 points for putting the answer correctly on stack
       MOVEM.L     (SP)+,D1/D3   ;get stuff off stack
*   1 point for cleaning up stack




       RTS
```

Name: _____ CWRUnet ID:_____

ANSWERS:

(a)



This part of the answer determined parts (b) and (c). Only the first stack, the input stack is necessary. This part is worth 8 points.

Commented code:

```
       ORG         $5000
main   MOVE.W      #6,D1          ;just for reference
       MOVE.W      #5,D2
       ADD         D1,D2
JSR    SUBR
A      DC.L        4              ;pass these parameters
B      DC.W        2
C      DS.W        1
* Your subroutine should return to the following
instruction.
DOIT   MOVE.W      (SP)+,C
       END         main

SUBR   MOVEM.L     D1/D3,-(SP)    ;save registers
*(b) put A into D1, B into D2

* WRONG ANSWER
       MOVE.W      (SP),D1        ;put A into D1
```

Name: _____ CWRUnet ID:_____

```
       MOVE.W     4(SP),D2      ;put B into D2
```

A lot of students will probably give the above answer which is wrong. The address for A is on the stack and the first instruction is correct but 4(SP) is the address of something else. You have to add the byte offset to the address of A to get the correct address..

* The correct answer is to put the return address into an address register and then use it with address register indirect addressing. This part is worth 6 points.

```
       MOVE.L     8(SP),A0
       MOVE.L     (A0)+,D1
*You could also increment A0 separately
       MOVE.W     (A0)+,D2
* A is technically a long word.  Must keep track of A0 to
get return address for (c).


INST   MOVE.L      #1,D3
       MULS        D1,D3          ;answer in D3
*(c) now put answer on stack

*    ANSWER
       MOVE.L      (SP),-2(SP)    ;move D1 down stack
       ADDQ.L      -2,SP          ;move SP down
       MOVE.L      6(SP),4(SP)    ;move D3 down
Put correct return address on stack.  The correct address
is the original RA+8 bytes.  There are several ways of
putting it on the stack.  The above code left A0 pointing
at C so I can add 2 bytes to A0 and put it on the stack as
the Return Address.
       ADDQ        #2,A0
       MOVE.L      A0,8(SP)

       MOVE.W      D3,$C(SP)      ;put answer in place
       MOVEM.L     (SP)+,D1/D3    ;restore registers
*   Several different answers are possible for this part.
This part is worth 6 points.


       RTS
```

(b) See above
(c) See above

Some comments on grading are in order.
The ordering of the registers due to the MOVEM instruction was worth 2 points.
The position of the stack pointer in the diagram was worth 1 point.
Not moving the stack back to make room for the answer was worth 5 points.
Forgetting the MOVEM in part (c) was worth 2 points.
Making the stack grow in the wrong direction was worth 2 points.
A long word answer for D3 was worth 1 point.

Name: _____ CWRUnet ID:_____

6. A subroutine SUB6 is called with parameters passed and returned on the stack.

```
        ORG         $6000
        MOVE.W      ARG1,-(SP)    ;push ARG onto stack
        MOVE.W      ARG2,-(SP)
        JSR         SUB6          ;call subroutine SUB2
        MOVE.W      (SP)+,RSLT    ;pop answer from stack
END6

ARG1    DC.W        4             ;base
ARG2    DC.W        2             ;exponent
RSLT    DS.W        1             ;result


SUB6    MOVE.W      xx(SP),D1     ;put ARG1 into D1
        MOVE.W      yy(SP),D2     ;put ARG2 into D2
        MOVE.L      #1,D3         ;put starting 1 into D3
LOOP6   SUBQ        #1,D2         ;decrement power
        BMI         EXIT          ;if D2-1<0 then quit SUB2
        MULS        D1,D3         ;multiply out
        BRA         LOOP6         ;and repeat as necessary

EXIT    MOVE.W      D3,zz(SP)
        MOVE.L      (d)           ;move return address to
                                  ;correct location for
                                  ;return
        ADDQ.L      (e)           ;increment SP to final
                                  ;value
        RTS
```

(a) What should be the value of xx to correctly retrieve ARG1 from the stack?
xx=_____ANSWER xx=+6 (6 points)

(b) What should be the value of yy to correctly retrieve ARG2 from the stack?
yy=_____ ANSWER xx=+4 (6 points)

(c) Specify the value of zz to properly put D3 on the stack so that it can be POPed from the stack and put into ARG3 AFTER the subroutine return.
zz=_____ ANSWER xx=+6 (6 points)

Specify the missing operand fields to make the subroutine work as described.

(d) _____ANSWER: (SP),2(SP) (4 points)
(e) _____ANSWER: #2,SP (4 points)

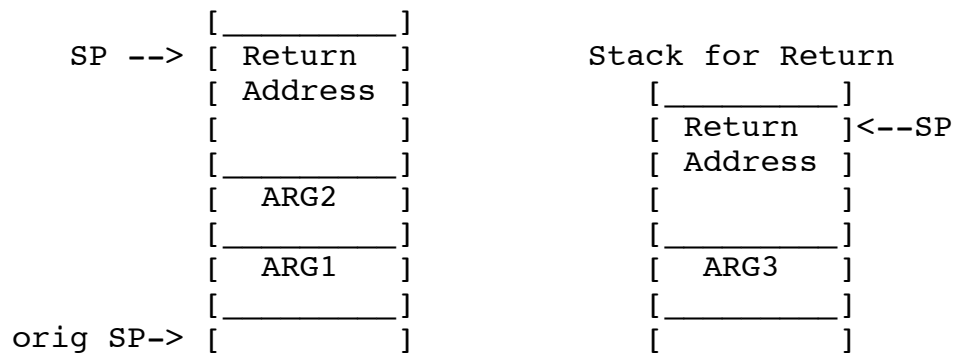
Name: _____ CWRUnet ID:_____

ANSWERS:
Commented program:

```
        ORG         $5000
        MOVE.L      ARG,-(SP)     ;push ARG onto stack
        MOVE.W      #4,D2         ;get another argument
        LEA         DATA2,A0
        PEA         (A0,D2.W)     ;push address onto stack
        JSR         SUB2          ;call subroutine SUB2
        MOVE.W      (SP)+,C2      ;pop answer from stack
END2

ARG     DC.L        4             ;base
B2      DC.W        2             ;exponent
C2      DS.W        1             ;result


SUB2    MOVE.L      xx(SP),D1     ;put ARG into D1

        MOVE.L      #1,D3         ;put starting 1 into D3
LOOP2   SUBQ        #1,D2         ;decrement power
        BMI         EXIT          ;if D2-1<0 then quit
                                  ;subroutine
        MULS        D1,D3         ;multiply out
        BRA         LOOP2         ;and repeat as necessary
EXIT    MOVE.W      D3,yy(SP)     ;put answer on stack on
                                  ;top of ARG
        MOVE.L      (c)(SP),6(SP)  ;move return address to
                                  ;correct location for
                                  ;return
        ADDQ.L      (d)#6,SP      ;increment SP to final
                                  ;value
        RTS
```

Name: _____ CWRUnet ID:_____

(a) The value of xx to correctly retrieve ARG1 from the stack is +6 See diagram below (6 points)



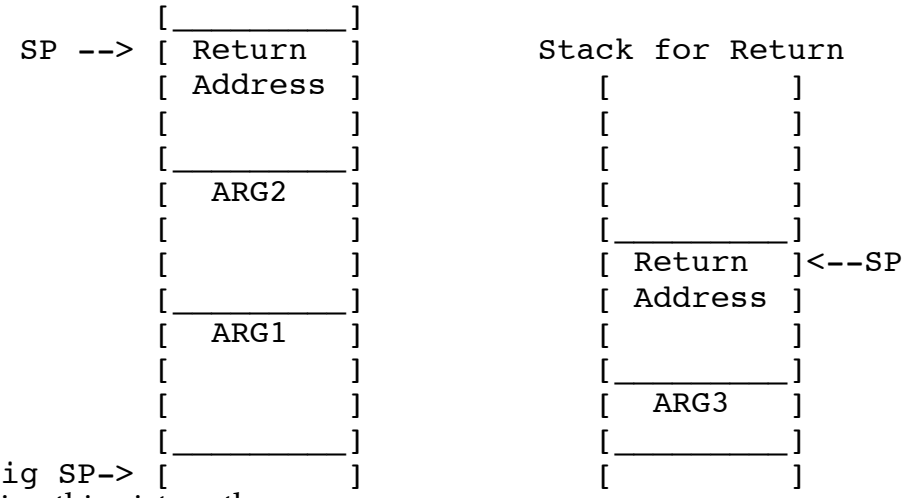
(b) The value of yy to correctly retrieve ARG2 from the stack is +4 See diagram above (6 points)

(c) The value of zz to properly put D3 onto the stack so that it can be POPed from the stack and put into ARG3 AFTER the subroutine return is +6 (6 points)

(d) (SP),2(SP) (4 points)
(e) #2,SP (4 points)

Name: _____ CWRUnet ID:_____

I have since discovered that some people used a long word arguement instead of a word length arguement. This makes the stack look like this:



Using this picture the answers are:

(a) The value of xx to correctly retrieve ARG1 from the stack is +8 See diagram below (6 points)

(b) The value of yy to correctly retrieve ARG2 from the stack is +4 See diagram above (6 points)

(c) The value of zz to properly put D3 onto the stack so that it can be POPed from the stack and put into ARG3 AFTER the subroutine return is +10 (6 points)

(d) (SP),6(SP) (4 points)

(e) #6,SP (4 points)