(1) Logical and arithmetic shift operations (may be combined with another problem)

2. What is in D1 after the following machine code executes?
(D1.L) = _____

```
MOVE.W    #$10B3,D1
MOVE.W    #24,D2
ASL.L     D2,D1
ASR.L     D2,D1
```

ANSWER:

(D1.L) = $FFFF FFB3

```
MOVE.W    #$10B3,D1  ;put data into D1
MOVE.W    #24,D2     ;put shift into D2, $18
ASL.L     D2,D1      ;shift to the left to clear
                     ;all but lower byte,
                     ;(D1)=$B3000000
ASR.L     D2,D1      ;now shift it back to the
                     ;right sign-extending all the
                     ;way, (D1)=$FFFFFFB3
```

2. What is output by the following program?

```
MASK     EQU     $000F
VALUE    DC.B    $5F
RESULT   DS.W    1

         LEA     RESULT,A1
         CLR.L   D0
         MOVE.B  VALUE,D0
         MOVE.L  D0,D1
         ROL.W   #4,D0
         LSR.B   #4,D0
         JSR     HexOut
```

EXPLANATION:

The number $0000050F is output.

```
         LEA     RESULT,A1  ;address where to put
                            ;result
         CLR.L   D0         ;clear register
                            ;(D0) = 0000 0000
         MOVE.B  VALUE,D0   ;get byte
                            ;(D0) = 0000 005F
         MOVE.L  D0,D1
         ROL.W   #4,D0      ;move byte to D0[4:11]
                            ;(D0) = 0000 05F0
         LSR.B   #4,D0      ;shift D0[4:7] to
                            ;D0[0:3]
                            ;(D0) = 0000 050F
         JSR     HexOut     ;print it
```

(2) program loops including DBcc and Bcc instructions

9. Consider the following program segment. You may assume that
(D0) = $FFFFFFFF,
(D1) = $FFFFFFFF and
(D2) = $FFFFFFFF
before the program segment is executed.

```
START  MOVE.W   #13,D1          ;A
       MOVE.W   #10,D2          ;B
       MOVEQ    #0,D0           ;set product to zero
       ANDI.L   #$0000FFFF,D1   ;set most significant
                                ;word of D1 to zero
ONE:   LSR.W    #1,D2           ;check LSB of B
       BCC.S    TWO             ;branch if zero was
found
       ADD.L    D1,D0           ;add if one
TWO:   LSL.L    #1,D1           ;shift multiply A left
one bit
       TST.W    D2              ;check if B is zero
       BNE.S    ONE             ;if not do it again
       TRAP     #0              ;quit
```

(a) Produce a pseudocode listing or a flowchart that explains what the above program does.

ANSWER:

```
This program does software multiplication of A
and B using a shift and add algorithm.
     D1.W <--A.              ;
     D2.W <--B.              ;multiplier
     D0 <--0.                ;product
     D1[15-31] <--0.         ;use masking to clear this
     Shift D2 right 1 bit.   ;move next bit of
                             ;multiplier into SR
ONE: if D2[0]=0 then         ;if the LSB was 1 then
     goto TWO                ;no product, skip it
     else                    ;else
     D0 <--D0+D1             ;add D1 to sum
TWO: Shift D1 left 1 bit.    ;multiply by 2
                             ;before summing
     if D2.W≠0 then          ;Any bits left in
multiplier?
     goto ONE.               ;If yes then repeat
```

(b) Specify what is in D0, D1, and D2 after the above program is executed.

(D0.L) = _____

ANSWER: (D0.L)=$ 00000082
(this is the product)

(D1.L) = _____

ANSWER: (D1.L)=$ 000000D0
(this was not altered by the program)

(D2.L) = _____

ANSWER: (D2.L)=$ FFFF0000
(it counted down)

9. Consider the following program segment:

```
INPUT     DC.B     %01010011
          MOVE.B   INPUT,D0
          BCLR     #7,D0
          MOVE.B   D0,D1
          MOVEQ    #0,D2
          MOVEQ    #0,D3
LOOP:     LSR.B    #1,D1      ;(a)
          ADDX.B   D3,D2
          TST.B    D1
          BNE.S    LOOP
          BTST     #0,D2      ;(b)
          BEQ      CLEANUP
          BSET     #7,D0
CLEANUP:
```

(a) What does the code beginning at (a) do, i.e. what is the function of the program?

(b) Specify what is in D0, D1, D2 and D3 when you reach the instruction labeled (b).
(D0.L) = _____
(D1.L) = _____
(D2.L) = _____
(D3.L) = _____

(c) What is in D0 after the above program is executed?
(D0.L) = _____

ANSWERS:

(a) What does the code beginning at (a) do, i.e. what is the function of the program?

This program performs a parity check almost like the one done in class. The byte to be checked is copied to D1, D2 is used to sum the 1's in INPUT, D3 is used as a dummy for an ADDX instruction. The bits are checked by shifting them to the right into the X bit of the SR. Then the bit is added to sum through the command ADDX.B D3,D2. Since D3 is always zero we are adding the X bit to D2 and summing the bits that are 1. Since 53 contains an even number of 1's, the result is that there is no change in the original number.

(b)
(D0.L) = $ xxxxxx53
(D1.L) = $ xxxxxx00
(D2.L) = $ 00000004
(D3.L) = $ 00000000

(c)
(D0.L) = $ xxxxxx53

3. The following code forms the sum of the one's complements of 20 word length numbers beginning at address ARR. Rewrite the code to be more efficient using postincrement addressing in A0 AND a DBxx instruction.

```
      ORG     $3000
      MOVEA.L #ARR,A0
      MOVEQ   #0,D1          ;word address
      MOVEQ   #20,D2         ;counter
      MOVEQ   #0,D0          ;sum in D0

LOOP: MOVE.W  (0,A0,D1.W),D3
      NOT.W   D3             ;complement
      ADD.W   D3,D0          ;add it
      ADDI.W  #2,D1          ;increment word address
      SUBQ.W  #1,D2          ;decrement counter
      BNE     LOOP

ANSWER:
      LEA     ARR,A0      ;could still remain MOVEA
      MOVEQ   #19,D2      ;use counter in D2 for
                          ;DBxx.  I looked carefully
                          ;for the use of 19, NOT 20
      MOVEQ   #0,D0       ;sum register
LOOP  MOVE.W  (A0)+,D3    ;increment addresses
      NOT.W   D3
      ADD.W   D3,D0
      DBRA    D2,LOOP     ;decrement and branch
```

The actual register contents after running this program are:

(D0)=$A1C0
(D1)=$0026
(D2)=$0000
(D3)=$EDCB

4. Consider the following program. The numbers in the table are SIGNED numbers.

```
      ORG     $400
ARR   DC.W    $0001, $23A2, $BAEE, $3400, $0000,
      DC.W    $2312, $FF23, $40FF, $22D1, $AB00
N     EQU     10

      ORG     $450
      LEA     ARR,A0
      MOVE.W  #N,D1
      MOVE.W  (A0)+,D0
      SUBQ    #1,D1
AGAIN CMP.W   (A0)+,D0
      BLE     GO
      MOVE.W  (-2,A0),D0
GO    DBRA    D1,AGAIN
```

(COMMENT: This program uses a BLE which will not be on the exam, but the rest of the problem is reasonable)

If (D0)=$AAFF1234, what is in (D0.L) AFTER the above program is executed.
(D0.L) = _____

ANSWER:

```
      ORG     $450
      LEA     ARR,A0      ;loads the table starting
                          ;address
      MOVE.W  #N,D1       ;load D1 with the table
                          ;index
      MOVE.W  (A0)+,D0    ;get the first table
                          ;element, put it in D0,
                          ;and increment the table
                          ;address
      SUBQ    #1,D1       ;decrement the table index
                          ;for the DBRA instruction
AGAIN CMP.W   (A0)+,D0    ;compare the next table
                          ;entry with that already
                          ;in D0, increment the
                          ;table address
      BLE     GO          ;signed branch, branch if
                          ;D0<=next table entry then
                          ;goto GO
      MOVE.W  (-2,A0),D0  ;otherwise, go back and
                          ;get the last entry and
                          ;put it in D0, DON'T
                          ;decrement address.  This
                          ;keeps the minimum entry
                          ;out of ARR which is $00
GO    DBRA    D1,AGAIN    ;repeat until entire table
                          ;is done
```

(D0.L) = $AAFFAB00 since $AB00 is the smallest element in the table. This is potentially confusing since negative numbers are smaller, i.e. less, than zero since a BLE was used. I would give a lot of partial credit for answering $AAFF0000

4. What are the values of A0, D1 and the Z bit of the CCR after the following program is executed?

```
        ORG     $4000
        MOVE    CHAR,D0
SEARCH  LEA.L   BUFFER,A0
        MOVE.W  #BUFSIZ,D1
        BRA     IN
SLOOP   CMP.B   (A0)+,D0
IN      DBEQ    D1,SLOOP
        RTS

(COMMENT: VERY GOOD PROBLEM  STUDY IT WELL!)



        ORG     $4100
BUFSIZ  EQU     8
BUFFER  DC.B    '0','E','E','A','P','2','8','2'
CHAR    DC.B    'A'
```

(D0.L) = _____
(A0.L) = _____
Z (of the CCR)=_____

ANSWER:

```
        ORG     $4000
* Find the first occurrence in BUFFER of the
* character in D0. Return with Z=0 if character not *
found, or with Z=1 and A0 pointing just past
* character if found.
        MOVE    CHAR,D0
SEARCH  LEA.L   BUFFER,A0   ;point to start of
                            ;buffer
        MOVE.W  #BUFSIZ,D1  ;get size of buffer
        BRA     IN          ;check for bufsize=0
SLOOP   CMP.B   (A0)+,D0    ;got a match?
IN      DBEQ    D1,SLOOP    ;fall through on match
                            ;or D1=-1
        RTS                 ;return Z=1 if char
                            ;found

        ORG     $4100
BUFSIZ  EQU     8
BUFFER  DC.B    '0','E','E','A','P','2','8','2'
CHAR    DC.B    'A'
```

(D0.L) = ASCII code for 'A'
(A0.L) = $0000 4104
Z (of the CCR) = 1

(1) math instructions: can include multiply, divide and extend instructions

7. Assume that (D0)=$FFFFFFFB and (D1)=$FFFFFFF6 before EACH of the following instructions is executed.
(a) What is in D0 and D1 after the instruction MULU D0,D1 is executed?
(D0.L) = _____
(D1.L) = _____

(b) What is in D0 and D1 after the instruction DIVS #5,D1 is executed?
(D0.L) = _____
(D1.L) = _____

ANSWERS:

(a)
(D0.L) = $FFFFFFFB
(D1.L) = $FFF10032 or 4293984306 decimal (only certain calculators will return such a large number)

(b)
(D0.L) = $FFFEFFFB
(D1.L) = $0000FFFE

This question is dividing -10 ($FFF6, the dividend) by 5 (the divisor). The result of this computation is -10/5 = -2 with a remainder of 0. These numbers are then placed into D1.L as shown above.

7. Assume that (D0)=$FFFFFFFB (-5) and (D1)=$0000001B (27) before EACH of the following instructions is executed.
(a) What is in D0 and D1 after the instruction MULU D0,D1 is executed?
(D0.L) = _____
(D1.L) = _____

(b) What is in D0 after the instruction DIVS D0,D1 is executed?
(D0.L) = _____
(D1.L) = _____

ANSWERS:
(a)
(D0.L) = $FFFFFFFB (-5)
(D1.L) = $001AFF79

(b)
(D0.L) = $FFFFFFFB
(D1.L) = $0002FFFB (2,-5)

This question is dividing 27 (the dividend) by -5 (the divisor). The result of this computation 27/-5 = -5 with a remainder of +2 which are placed into D1.L as shown above.

5.Consider the following program segment:

```
     MOVE.L   #NUMBER,D0
     DIVU     #3,D0
```

(a) If NUMBER EQU $50005533 what is the result of the instruction?
(D0).L = _____

(b) If NUMBER EQU $00010000 what is the result of the instruction?
(D0).L = _____

ANSWERS:
(a) (D0) = $50005533 An overflow is detected because of the size of the operand. D0 is not changed.

(b) (D0) = $00015555 This number successfully divides. The quotient is $0001 with a remainder of $5555

5. The following program processes signed words beginning at DATA. What is in D3 after the following program is executed?

```
        ORG     $5100
N       EQU     3
DATA    DC.W    $FFF0,$0002,$000A

        ORG     $5000
START   CLR.L   D1          ;clear register
        MOVE    #N,D1
        LEA     DATA,A1
        CLR.L   D2
        CLR.L   D3
        MOVE.L  D1,D4
        SUBQ.W  #1,D4
LOOP2   MOVE.W  (A1)+,D2
        ADD.W   D2,D3
        DBRA    D4,LOOP2
        EXT.L   D3
        DIVS    D1,D3
        END
```

(D3.L) = _____

Why is the EXT.L instruction necessary before the DIVS?

ANSWER:

```
        ORG     $5100
N       EQU     3
DATA    DC.W    $FFF0,$0002,$000A

        ORG     $5000
START   CLR.L   D1          ;clear register
        MOVE    #N,D1       ;number of numbers,
                            ;(D1)=0000 0003
        LEA     DATA,A1     ;address of numbers
        CLR.L   D2          ;clear upper half of D2
        CLR.L   D3          ;clear sum
        MOVE.L  D1,D4       ;set counter
        SUBQ.W  #1,D4       ;to N-1
LOOP2   MOVE.W  (A1)+,D2    ;get number
        ADD.W   D2,D3       ;sum it
        DBRA    D4,LOOP2    ;do it till N-1
        EXT.L   D3
        DIVS    D1,D3
        END
```

(D3.L) = _____ANSWER: $FFFF FFFF

The format of the answer is [remainder|quotient]
The program computes the average of the N words beginning at DATA. In this case the program sums $FFF0 (-16), $0002 and $000A to get $FFFFFFFC (-4) in D3. The average is computed by the DIVS which divides $FFFFFFFC (-4) by N (3) to get $FFFF (-1) with a remainder of $FFFF (-1). Note that the remainder has the same sign as the dividend, i.e. negative.

The EXT.L instruction is necessary before the DIVS because we are dealing with word length signed numbers. A DIVS assumes that the dividend will be a signed long word; hence, D3 has to be sign extended to get the correct signed answer.

1. The following instructions are executed sequentially. What is in D0, D2 and D3 after this program segment is executed?

```
      ORG       $1000           ;problem 1
      CLR.L     D0              ;clear the register
      MOVE.W    #$D200,D0       ;put dividend in
      MOVE.W    D0,D2           ;duplicate the dividend
(word)
      MOVE.L    #$00020145,D1   ;divisor
      DIVU      D1,D0           ;unsigned divide
      EXT.L     D2              ;preserve the sign
      MOVE.L    D2,D3           ;make problem more
interestng
      DIVS      D1,D3           ;signed divide
```

(D0.L) = _____
(D2.L) = _____
(D3.L) = _____

ANSWERS:
The first divide is an unsigned divide. The quotient is put in the lower half of D0, the remainder is put in the upper half. In decimal:$0000D200 = 53,760 and $0145 = 325

The division of 53760 by 325 gives 165 with a remainder of 135, or in hex,$00A5 with a remainder of $0087. The result would then be (D0.L) =$008700A5

The EXT sign extends $D200 to a long word. So, (D2.L)=$FFFFD200.

The second divide is a signed divide. The quotient is put in the lower half of D0, the remainder is put in the upper half. In decimal: $FFFFD200 = -11,776 and $0145 = 325 as before

The division gives -36 with a remainder of -76, or in hex, $FFDC with a remainder of $FFB4. The result would then be (D0.L) =$ FFB4FFDC

(3) basic operation of stacks and subroutines

14. Insert the appropriate code after RETURN: to allow the subroutine to restore registers and properly return to the next executable instruction.

```
          MOVE.W     #3,LIST2
          JSR        PRINTIT
          DC.L       LIST2
          MOVE.W     #4,LIST2
          <rest of program>

PRINTIT:  MOVE.L     A6,-(SP)
          MOVEA.L    SP,A6
          MOVEM.L    D0-D1/A0-A1,-(SP)
RETURN:   <code to do something goes here>
* begin your code here
* <your code>
          RTS

LIST2:    DS.L       1
          END
```

ANSWER:
The stack looks like this AFTER the MOVEM instruction.:

```
SP-->   [      D0      ]
        [      D1      ]
        [      A0      ]
        [      A1      ]
        [      A6      ]
        [Return Address]
```

The stack width is shown as long word width here for convenience.

A solution for the code in PRINTIT is:

```
PRINTIT:  MOVE.L     A6,-(SP)
          MOVEA.L    SP,A6
          MOVEM.L    D0-D1/A0-A1,-(SP)
RETURN:
* The following three commands properly manipulate
* the stack.
          MOVEM.L    (SP)+,D0-D1/A0-A1  ;pop the saved
                                        ;registers
                                        ;off the stack
          ADDA.L     #4,SP              ;flush the
                                        ;value of
                                        ;A6 put on
                                        ;the stack
          ADD.L      #4,(SP)            ;increment
                                        ;the return
                                        ;address to
                                        ;return past
                                        ;the DC.L
          RTS
```

A number of clever people did
MOVEM.L   (SP)+,D0-D1/A0-A1/A6
which is perfectly fine.
Scoring: -4 points for missing the return address.

15. The following program fragment pushes two word length variables onto the stack as input to the subroutine TEST shown below. The subroutine returns one word length output on the stack.

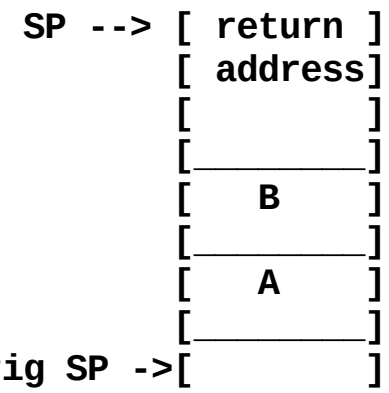
The correct code for using TEST is shown below:

```
GO:     MOVE.W     A,-(SP)     ;pass A to test
        MOVE.W     B,-(SP)     ;pass B to test
        JSR        TEST
        MOVE.W     (SP)+,C     ;get C from test
```

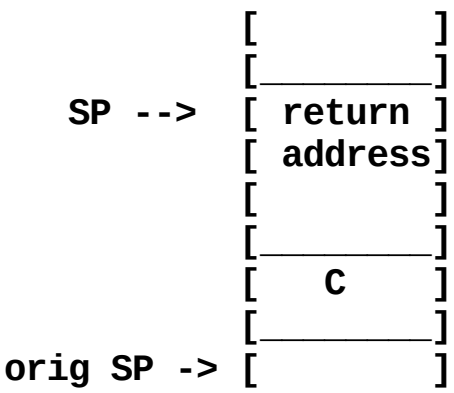
You are to provide the requested information for subroutine TEST.

```
* (a) Finish the next two instructions to correctly
*     get * A and B from the stack

TEST    MOVE.W     _____,D1    ;get A
        MOVE.W     _____,D2`   ;get B
* code here computes something,result is in D3.W
IT:
* (b) Put your instructions here to properly return
*     from the subroutine


        RTS
ANSWERS
* (a)
TEST    MOVE.W     6(SP),D1    ;get A
        MOVE.W     4(SP),D2`   ;get B

* (b) Put your instructions here to properly return
*     from the subroutine
        MOVE.W     D3,6(SP)    ;put C on top
                               ;of A
        MOVE.L     (SP),2(SP)  ;move return address
                               ;down two bytes
        ADDA       #2,SP       ;move SP down two
                               ;bytes
        RTS
```

immediately after the subroutine call:



what the answer should do:



These stacks are shown as byte width. Note that the solution places C on the stack in place of A, moves the SP up in memory 2 bytes, and changes the SP to the new value.

16. The user stack is as shown below. The instruction

```
     MOVEM.L     (SP)+,D4/A0/D2/A6/A5
```

is executed. Indicate the values of of the indicated registers.

```
SP -->   [$1111]
         [$0000]
         [$FFFF]
         [$1234]
         [$0002]
         [$0040]
         [$FFFC]
         [$0001]
         [$A021]
         [$0100]
         [$0002]
         [     ]
         [     ]
         [     ]
```

ANSWER:

(A0.L) = $00020040

(A5.L) = $FFFC0001

(A6.L) = $A0210100

(D2.L) = $11110000

(D4.L) = $FFFF1234

17. Given the program segment shown below, answer the following questions:

```
        MOVEM.L   D0/A0,-(SP)
A:      MOVE.W    16(SP),D0   ;get input
        ADD.W     D0,D0
        MOVEA.L   12(SP),A0
        MOVE.W    D0,(A0)     ;save output
        MOVE.W    (SP),10(SP)
        MOVEM.L   (SP)+,D0/A0
        ADDA.L    #10,SP
        RTS
```

```
SP -->  [$0000]
        [$000A]
        [$0000]
        [$2000]
        [$0672]
        [$1000]
        [$0000]
        [$2020]
        [$0012]
        [$0000]
        [$12E8]
        [$1020]
```

If the stack is as shown above just before the instruction labeled A is executed:

(a) What is the value of the input to the subroutine?
(b) What value will the subroutine output for the input in (a)?
(c) Where is the output put? (be specific)
(d) To what address will the subroutine return

ANSWERS:

```
*       A commented program helps dramatically
        MOVEM.L   D0/A0,-(SP)
A:      MOVE.W    16(SP),D0    ;get input,$0012
        ADD.W     D0,D0        ;double input
        MOVEA.L   12(SP),A0    ;get $00002020
                               ;put in A0
        MOVE.W    D0,(A0)      ;save output
                               ;to $00002020
        MOVE.W    (SP),10(SP)  ;replace $1000 by
                               ;$0000
        MOVEM.L   (SP)+,D0/A0  ;restore registers
        ADDA.L    #10,SP       ;point to $000012E8
        RTS

SP -->  [$0000]  these two
        [$000A]  are D0
        [$0000]  these two
        [$2000]  are A0
        [$0672]
```

(a) $0012 SCORING: -1 if off by one word
(b) $0024
(c) ($00002020.W)=$0024
(d) $000012E8

14. What, if anything, is wrong with the following program?

```
       ORG     $1000       ;main program
MAIN   NOP
       JSR     SUB         ;call subroutine
       MOVE.W  (SP)+,D0    ;get result from stack
       <more code>         ;more instructions here

       ORG     $2000       ;subroutine
SUB    MOVE.W  #$01,-(SP)  ;place result on stack
       RTS                 ;return from subroutine

       END     MAIN
```

ANSWER:
The main program is expecting a result to be on the stack AFTER the return address.

```
SP---> [ return  ]
       [ address ]
```

However, the subroutine puts #01 on the stack like this.

```
SP---> [   $01   ]
       [ return  ]
       [ address ]
```

The RTS will use the top of the stack as the return address, but the top of the stack contains the subroutine result, which will result in an incorrect return address.

15. Assume (A2) = $053F0A, (D2)=$0004, and (D1)=$FF00. Show the contents of memory after the following instruction is executed.

```
     MOVEM.W     D1-D2/A2,(A2)

($53EFE)=$ 0A     ($53F07)=$ AA
($53EFF)=$ EE     ($53F08)=$ EE
($53F00)=$ 03     ($53F09)=$ AB
($53F01)=$ 82     ($53F0A)=$ 02
($53F02)=$ 0A     ($53F0B)=$ 82
($53F03)=$ EE     ($53F0C)=$ 12
($53F04)=$ 30     ($53F0D)=$ 00
($53F05)=$ 00     ($53F0E)=$ BC
($53F06)=$ FF     ($53F0F)=$ 2D
```

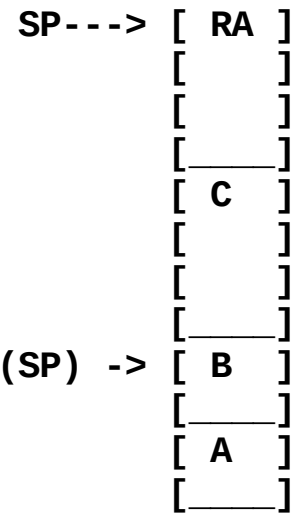
ANSWER:
The register (word length contents) are put in memory in post increment form in the order D1, D2, A2.This results in the following memory changes.

```
($53EFE)=$ 0A     ($53F07)=$ AA
($53EFF)=$ EE     ($53F08)=$ EE
($53F00)=$ 03     ($53F09)=$ AB
($53F01)=$ 82     ($53F0A)=$ 02 <---$FF
($53F02)=$ 0A     ($53F0B)=$ 82 <---$00
($53F03)=$ EE     ($53F0C)=$ 12 <---$00
($53F04)=$ 30     ($53F0D)=$ 00 <---$04
($53F05)=$ 00     ($53F0E)=$ BC <---$3F
($53F06)=$ FF     ($53F0F)=$ 2D <---$0A
```

16. The following program fragment pushes two word length variables onto the stack as input to the subroutine COMPUTE shown below. What are the necessary values for M, N, ADDRC and PARAM for this subroutine to properly work as shown.

```
* PROGRAM BEGINS HERE
MAIN    MOVEQ    #4,D2
        MOVE.L   #$34,D5
        MOVE.L   #$FFA2,D6
        MOVE.W   D5,-(SP)
        MOVE.W   D6,-(SP)
        PEA      C
        JSR      COMPUTE
        MOVE.L   C,D7          ;do something with C
        TRAP     #0            ;stop program

M       EQU      ?             ;what are the values
                               ;of these constants?
N       EQU      ?
ADDRC   EQU      ?
PARAM   EQU      ?
PARAM2  EQU      ?

COMPUTE
        MOVE.W   (M,A7),D0
        EXT.L    D0
        MOVE.W   (N,A7),D1
        EXT.L    D1
        MOVE.L   (ADDRC,A7),A0
        ASL.L    #1,D0
        ADD.L    D1,D0
        MOVE.L   D0,(A0)
        MOVE.L   (SP),(PARAM,SP)
        ADD.L    #PARAM2,SP
        RTS

C       DS.L     1

        END
```

ANSWER: The stack looks like this inside the subroutine COMPUTE:



The correct answers to properly retrieve the data are:

M	**10**
N	**8**
ADDRC	**4**
PARAM	**8**
PARAM2	**8**

The fully commented program is:

```
* PROGRAM BEGINS HERE
MAIN    MOVEQ    #4,D2
        MOVE.L   #$34,D5     ;get data
        MOVE.L   #$FFA2,D6   ;now get more data
        MOVE.W   D5,-(SP)    ;push data A on stack
        MOVE.W   D6,-(SP)    ;push data B on stack
        PEA      C           ;put address of
                             ;C on stack
        JSR      COMPUTE
        MOVE.L   C,D7        ;do something with C
        TRAP     #0          ;stop program

M       EQU      10          ;what are the values
                             ;of these constants?
N       EQU      8
ADDRC   EQU      4
PARAM   EQU      8
PARAM2  EQU      8

COMPUTE
        MOVE.W   (M,A7),D0        ;get A
        EXT.L    D0
        MOVE.W   (N,A7),D1        ;get B
        EXT.L    D1
        MOVE.L   (ADDRC,A7),A0   ;get address of C
        ASL.L    #1,D0            ;compute M*2
        ADD.L    D1,D0            ;add N
        MOVE.L   D0,(A0)          ;save to memory
        MOVE.L   (SP),(PARAM,SP) ;clear stack
        ADD.L    #PARAM2,SP       ;flush the stack
        RTS

C       DS.L     1

        END
```

17. The subroutine SWAP takes the addresses of two long words passed by registers and adds the word length contents of their addresses according to the following program fragment:

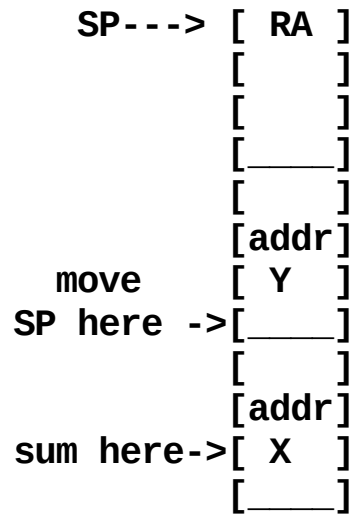
```
START:  LEA     X,A0          ;pass X and Y
                              ;by reference
        LEA     Y,A1
        JSR     SWAP          ;add them in D2
        MOVE.W  D2,D0         ;put output into D0
        JSR     HEXOUT_LONG   ;print sum
        TRAP    #0            ;it's all over

SWAP    MOVE.W  (A0),D2
        MOVE.W  (A1),D3       ;get Y
        ADD.W   D2,D3
        MOVE.W  D3,(A1)       ;put sum into Y
        RTS

X       DC.W    12
Y       DC.W    5
        END     START
```

Rewrite the subroutine so that all input and output parameters are passed on the stack as shown below.

```
START:  MOVE.L  #X,-(SP)      ;pass X and Y by reference
        MOVE.L  #Y,-(SP)      ;could also use PEA
        JSR     SWAPPER       ;add them
        MOVE.W  (SP)+,D0
        JSR     HEXOUT_LONG   ;print it out
        TRAP    #0            ;it's all over

SWAPPER
*       your code goes here
        RTS

X       DS.L     1
Y       DS.L     1
        END      START
```

ANSWER:

The stack inside the subroutine looks like this.



Code to properly manipulate the stack:

```
SWAPPER:
        MOVEA.L  4(SP),A1    ;get address of X
        MOVEA.L  8(SP),A2    ;get address of Y
        MOVE.W   (A1),D2     ;now get X
        ADD.W    (A2),D2     ;now compute X+Y
        MOVE.W   D2,10(SP)   ;move answer.W to
                             ;bottom word of
                             ;X address
        MOVE.L   (SP),6(SP)  ;move return address
                             ;to just below sum
        ADD.L    #6,SP       ;flush stack
        RTS
```

4. Answer the questions about the program given below.

```
       ORG      $4000
SUBR   MOVEM.L  A0/A3/D2,-(SP)
       ADDA     #$8,SP
       MOVE.L   (SP)+,A1
       ADD.L    (SP)+,A2
B:     NOP                          ;(a)
       ADDA.W   #OFFSET,SP
       MOVEM.L  (SP)+,A3/D2/A0
       RTS
```

(a) Draw a picture of what is on the stack at (a) beginning with the return address (assuming SUBR has just been called). Be sure to specify the width of the stack and your SP on your drawing.

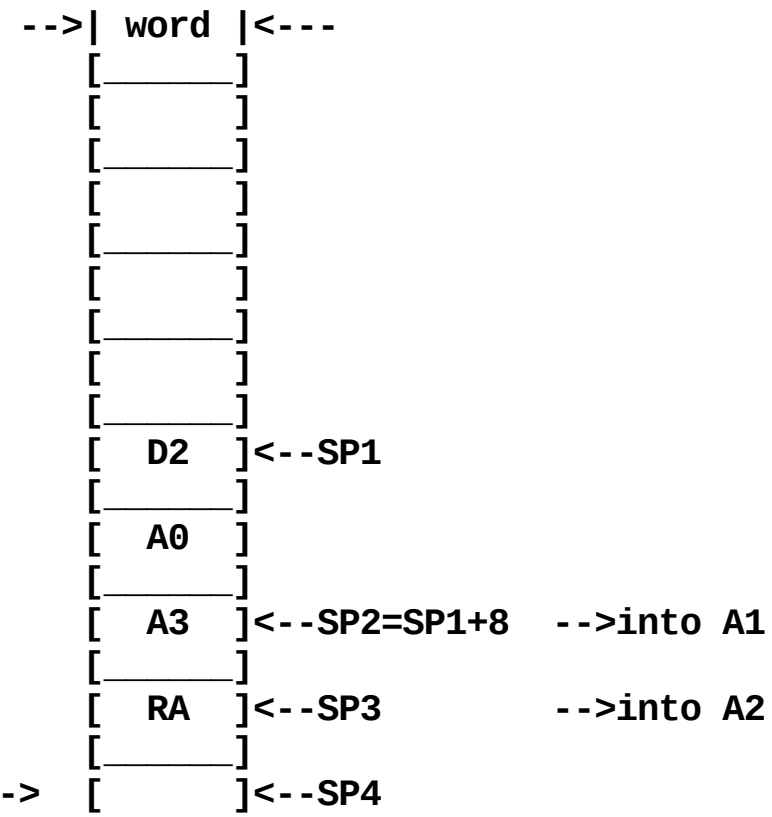
```
        [      ]
        [      ]
        [      ]
        [      ]
        [      ]
        [      ]
        [      ]
        [      ]
        [      ]
        [      ]
        [      ]
        [      ]
        [      ]
        [      ]
        [      ]
        [      ]
        [      ]
        [      ]
SP-->   [      ]
```

(b) Specify OFFSET (a signed hex number) so that the MOVEM and RTS instructions correctly function when the subroutine returns.

ANSWER:

```
      ORG       $4000
SUBR  MOVEM.L   A0/A3/D2,-(SP)  ;SP1
      ADDA      #$8,SP           ;SP2
      MOVE.L    (SP)+,A1         ;SP3, gets A3 from stack
                                 ;puts into A1
      ADD.L     (SP)+,A2         ;SP4, gets RA from stack
                                 ;puts into A2
B:    NOP                        ;(a)
      ADDA.W    #OFFSET,SP       ;
      MOVEM.L   (SP)+,A3/D2/A0   ;
      RTS
```

(a) In predecrement (push) mode the registers are ALWAYS pushed on the stack in the order A7,A6,...,A2,A1,D7,D6,...,D1.



(b) The SP is at the original value but needs to go -16 bytes to properly pop D2, etc. off the stack. Hence, OFFSET = - 16 (decimal) = $FFF0

5. A subroutine SUB2 is called with parameters passed and returned on the stack.

```
        ORG       $5000
        MOVE.L    ARG,-(SP)     ;push ARG onto stack
        MOVE.W    #4,D2
        LEA       DATA2,A0
        PEA       (A0,D2.W)
        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)
        MOVE.L    (c)           ;move return address to
                                ;correct location for
                                ;return
        ADDQ.L    (d)           ;increment SP to final
                                ;value
        RTS
```

(a) What should be the value of xx to correctly retrieve ARG from the stack?
xx=_____
(b) Specify the value of yy to properly put D3 on the stack so that it can be POPed from the stack and put into C2 AFTER the subroutine return.
yy=_____

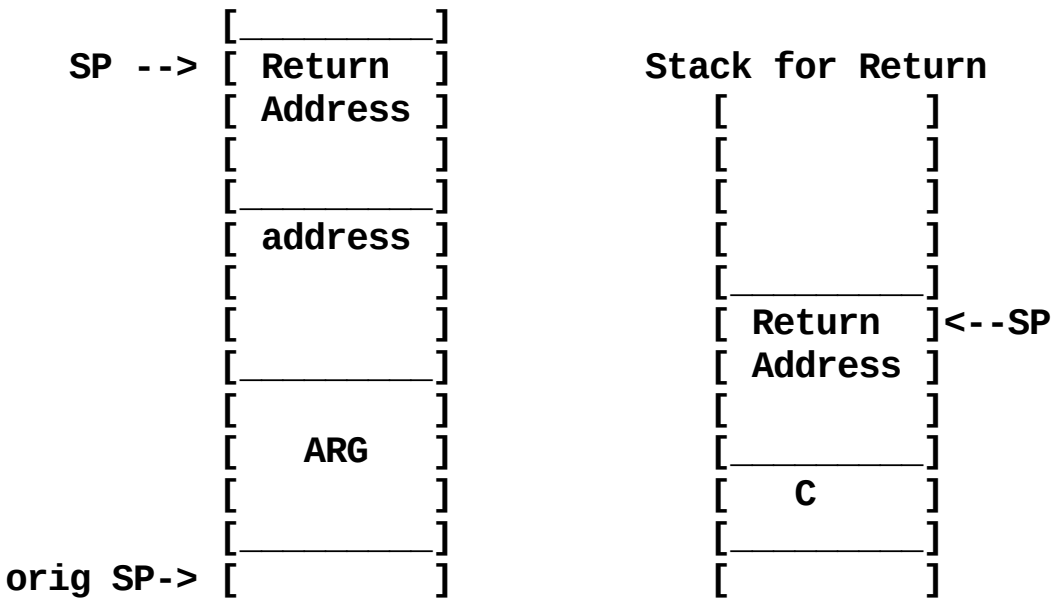
Specify the missing operand fields to make the subroutine work as described.

(c) _____
(d) _____

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
```

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



(b) The value of yy to properly put D3 on the stack so that it can be POPed from the stack and put into C2 AFTER the subroutine return is $0A (6 points)

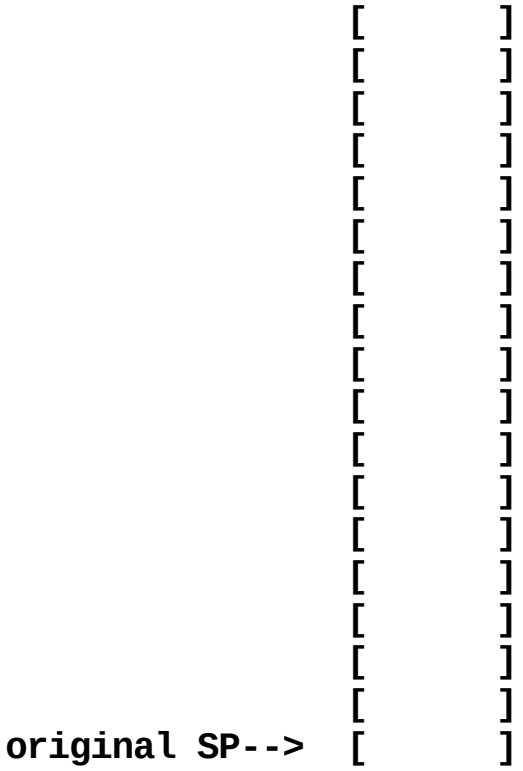
(c) (SP),6(SP) (4 points)
(d) #6,SP (4 points)

The most common answers were:
(a) 8
(b) 8
(c) (SP),4(SP)
(d) #4,SP

6. A student has decided to use in line coding of data to pass parameters to a subroutine. The main program shown below calls the subroutine SUBR. The stack pointer is initially at $8000. Answer the following questions.

```
        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.
DOIT    MOVE.W      (SP)+,C
        END         main

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



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



        RTS
```

(a) What is on the stack when the PC is at the label INST? Explicitly show all stack contents AND addresses. The inital SP (before the program is executed) is shown.



(b) What instruction(s) must go into the box to put the word at A into D1 and the word at B into D2?
(c) What instruction(s) must go into the box such that the subroutine will return the answer in D3 onto the stack such that it it popped off the stack at DOIT.

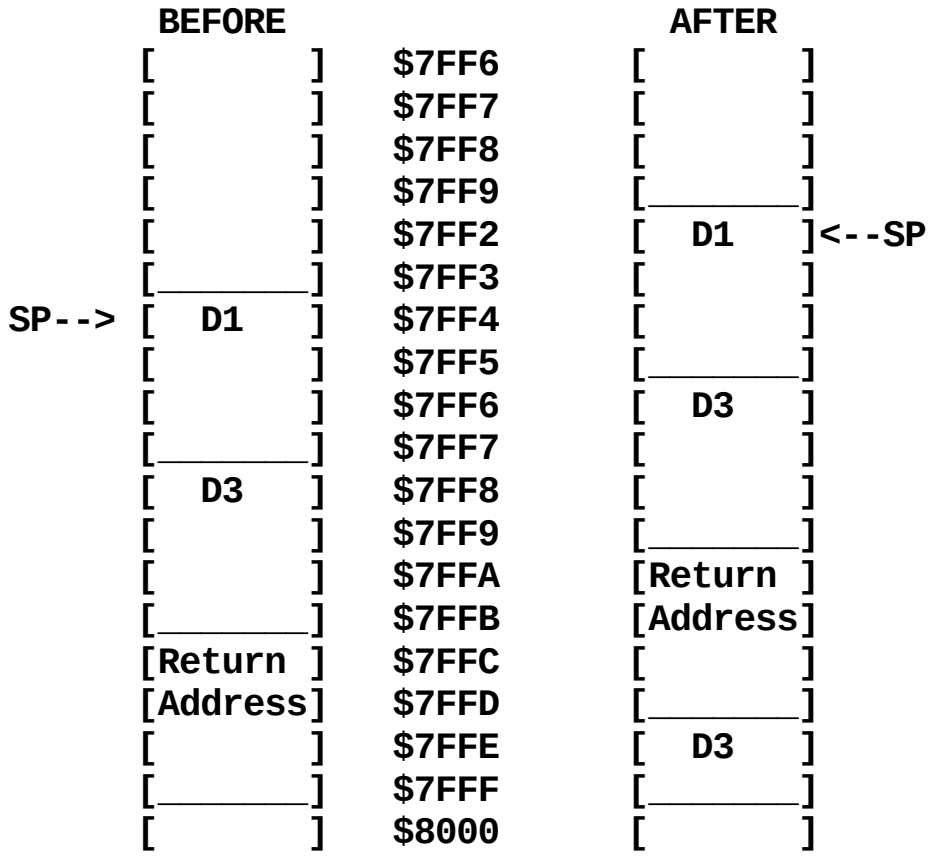
ANSWERS:
Commented code:

```
        ORG         $6000
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

* ANSWER
        MOVE.W      (SP),D1         ;put A into D1
        MOVE.W      4(SP),D2        ;put B into D2
```

A lot of students gave 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..

```
* There were a large number of unanticipated ways in
* which students inserted the correct A and B into D1
* and D2 respectively.  We gave full credit for these
* unexpected methods.  This part was worth 6 points.
*  UNEXPECTED ANSWERS
        MOVE        #A,D1
        MOVE        #B,D2

        MOVE        A,D1
        MOVE        B,D2


        MOVE.L      8(SP),A0
        MOVE.L      (A0)+,D1
        MOVE.W      (A0),D2

* A was technically a long word but we did not take
off any points for that.


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
        MOVE.L      $A(SP),8(SP)    ;move RA down
        MOVE.W      D3,$C(SP)       ;put answer in place
        MOVEM.L     (SP)+,D1/D3     ;restore registers
*     Several answers were possible. This part was
worth
*     6 points.


        RTS
```

(a)



This part of the answer determined parts (b) and (c) and was worth 8 points.

(b) See above
(c) See above

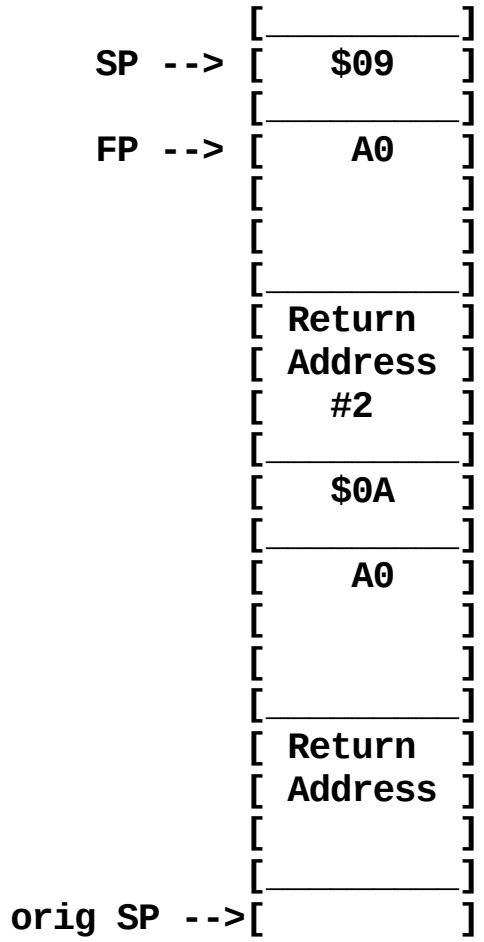
Some comments on grading are in order.
The ordering of the registers due to the MOVEM instruction was worth 2 points.
The position of the stack pointer in the diagram was worth 1 point.
Not moving the stack back to make room for the answer was worth 5 points.
Forgetting the MOVEM in part (c) was worth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

Probably NOT on this exam:

7. Consider the recursive routine FACTOR. What are the contents of the stack and A0 after the first TWO (2) calls of the subroutine. You may assume that (SP)=$8000 when the program begins execution.

```
DATAX    EQU      $7600
PROGRAM  EQU      $7000
         ORG      DATAX
NUMB     DC.W     $A          ;number
F_NUMB   DS.W     1           ;answer

         ORG      PROGRAM
MAIN     MOVE.W   NUMB,D0     ;get number
         JSR      FACTOR      ;compute
         MOVE.W   D0,F_NUMB   ;store answer
         TRAP     #0

FACTOR   LINK     A0,#-2
         MOVE.W   D0,-2(A0)
         SUBQ.W   #1,D0
         BNE      F_CONT
         MOVEQ    #1,D0
         BRA      RETURN
F_CONT   JSR      FACTOR
         MULU     -2(A0),D0
RETURN   UNLK     A0
         RTS

DONEIT   END
```

```
ANSWER: Commented program:
DATAX    EQU      $7600
PROGRAM  EQU      $7000
         ORG      DATAX
NUMB     DC.W     $A          ;number
F_NUMB   DS.W     1           ;answer, factorial of
number

         ORG      PROGRAM
MAIN     MOVE.W   NUMB,D0     ;get number
         JSR      FACTOR      ;compute
         MOVE.W   D0,F_NUMB   ;store answer
         TRAP     #0

FACTOR   LINK     A0,#-2
         MOVE.W   D0,-2(A0)
         SUBQ.W   #1,D0        ;decrement number
         BNE      F_CONT       ;not end of factorial
process
         MOVEQ    #1,D0        ;factorial:=1
         BRA      RETURN
F_CONT   JSR      FACTOR       ;continue factorial
process
         MULU     -2(A0),D0    ;factorial:=N*(N-1)
RETURN   UNLK     A0
         RTS
DONEIT   END
```



The location of the SP and FP were worth 2 points each. The length (size) and content of each item on the stack were worth 1 point each.