EEAP 282

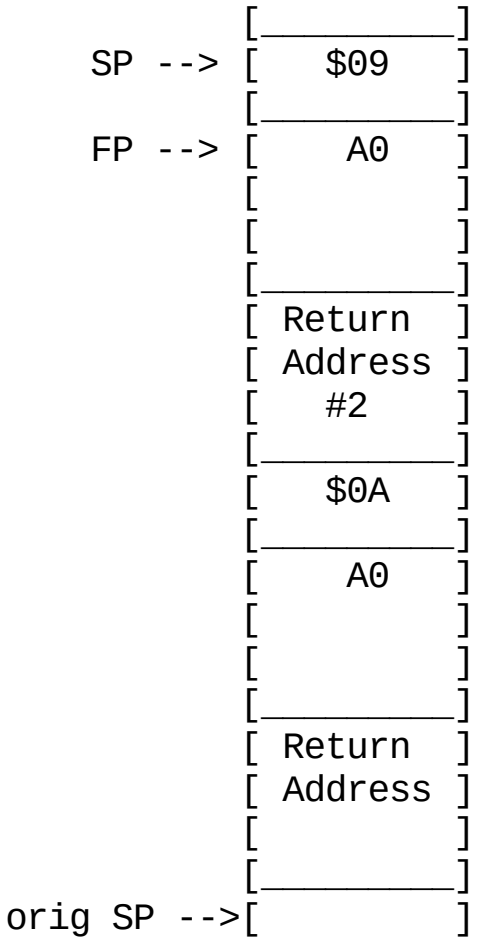
Exam #4 Sampler

LINK/UNLK

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.

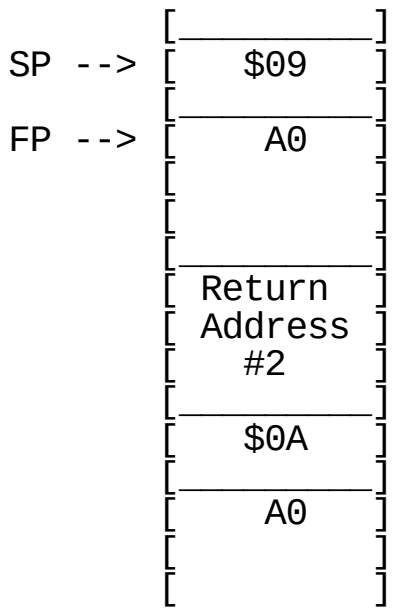
7. Consider the recursive routine FACTOR. What are the contents of the stack and A0 after the first TWO (2) calls of the subroutine. You may assume that (SP)=$8000 when the program begins execution.

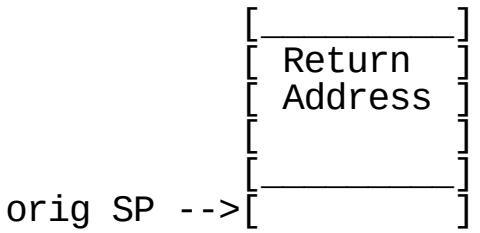
```
DATAX     EQU       $7600
PROGRAM   EQU       $7000
          ORG       DATAX
NUMB      DC.W      $A          ;number
F_NUMB    DS.W      1           ;answer

          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.

14. (Lawson 282) This program calls a subroutine using LINK and UNLK instructions. What is on the stack after

(a) the instruction JSR is executed
(b) the instruction LINK is executed
(c) the instruction UNLK is executed
(d) the instruction RTS is executed

```
            ORG     $1000
N           EQU     8
M           EQU     8

            ADD.L   #-N,SP
            MOVE.L  ARG,-(SP)
            PEA     X
            JSR     SUBR
            ADD.L   #8,SP
            MOVE.L  (SP)+,D1
            MOVE.L  (SP)+,D2
ARG         DC.L    $01234567
X           DS.B    200

SUBRLINK    A1,#-M
            MOVE.L  LOCAL1,-4(A1)
            MOVE.L  LOCAL2,-8(A1)
            ADD.L   #1,-4(A1)
            MOVEA.L 8(A1),A2
            MOVE.L  OUTPUT1,16(A1)
            UNLK    A1
            RTS

LOCAL1      DC.L    $98765432
LOCAL2      DC.L    $87654321
OUTPUT1     DC.L    'ABCD'

            END
```

INTERRUPTS & EXCEPTIONS:

12. The following code is executed with (SR)=$2000, (USP)=$4000, (SSP)=$7000, and (D2.L)=$00008000. The system has 32K ($8000) of memory, i.e. there is no memory at any address such as $8001 which is greater than $8000.

```
                                    ORG     $5500
00005500 21FC 0000 551C    BEGIN:   MOVE.L  #ROUT,$08
            0008
00005508 4FF9 0000 8000             LEA     $8000,SP        ;set SSP to
32k ($8000)
0000550E 3F7C 0007 0006             MOVE.W  #7,-(SP)        ;move back to
$7FFE
00005514 2EF8 4000                  MOVE.L  $4000,(SP)+     ;move forward
                                                   to >32k, generates bus
                                                   error exception
00005518 0642 0045                  ADD     #$45,D2         ;generic
                                                   instruction and rest
                                                   of program

                           ROUT:                            ;bus error
service routine
0000551C 4FEF 001A                  LEA     $8010,SP        ;put new
                                                   value into SP, does
                                                   not cause bus error
00005520 4E73                       RTE                     ;program
                                                   return, causes bus
                                                   error since address is
                                                   greater than $8000,
                                                   68000 HALTS
```

What happens when the program is executed? Be sure to state what, if any, exceptions occur, where they occur and why they occur.

12. When you execute the instruction TRAP #0, where (i.e. at what address) does the 68000 expect to find a service routine?

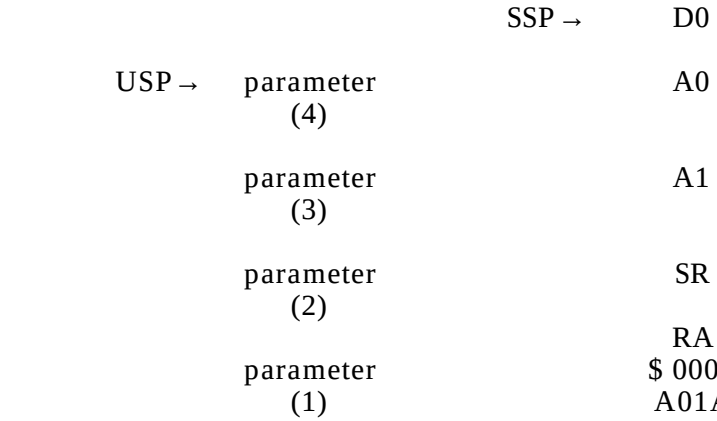
ANSWER: The exception vector number is 32+0. According to the rules for exceptions, the exception vector table address is then $32x4=$80

10. The following program is assembled and loaded at address $A000. The programmer calls a TRAP #1 exception which is to access the parameters placed on the stack by his/her program. Assume the program starts in USER mode.

```
                          * THIS IS THE USER'S PROGRAM
                                 ORG     $A000
                          START2:
                          * put input parameters on stack
0000A000 2F3C 0000 0003          MOVE.L  #3,-(SP)        ;parameter 1
0000A006 2F3C 0000 A500          MOVE.L  #BUF,-(SP)      ;parameter 2
0000A00C 2F3C 0000 0200          MOVE.L  #512,-(SP)      ;parameter 3
0000A012 2F3C 0000 0002          MOVE.L  #2,-(SP)        ;parameter 4
0000A018 4E41                    TRAP    #1
0000A01A 4E71                    NOP

                                 ORG     $A500
0000A500                  BUF:   DS.B    512
                          * service routine for TRAP #1.
                                 ORG     $A700
                          TRAP1:
0000A700 48E7 80C0               MOVEM.L D0/A0-A1,-(SP)
                          STACKS:
                          * code for processing TRAP goes here

                          * instructions for part (b) go here
0000A004 4E69                    MOVE.L  USP,A1          ;A1 points at
uk

0000A006 2029 0008               MOVE.L  8(A1),D0        ;get data
with P
0000A00A 4CDF 0301               MOVEM.L (SP)+,D0/A0-A1
0000A00E 4E73                    RTE
                                 END
```

(a) Draw a picture of the system and user stacks at the point labeled STACKS: in the TRAP #1 service routine.

ANSWER:



Note: these stacks are shown as word width instead of the usual byte width for brevity.

(b) Immediately after the label STACKS: in the TRAP #1 service routine the programmer wants to access the data labeled parameter

2 that was put on the stack in the user program and put it into D0. Give 68000 code for doing this.

ANSWER:

```
            MOVE.L  USP,A1          ;A1 points at user
                                    stack
            MOVE.L  8(A1),D0        ;get data with
                                    offset of 2 long
                                    words
```

11. At START: the system stack pointer is $A000 and the status register is $2000. The exception vector table has been loaded with the addresses of all appropriate service routines. Assume that all service routines return to the next line. The following program segment is executed.

```
START:      LEA       $8000,SP              ;1
            MOVE.L    #$9000,$700D          ;2
            DC.W      $FFFF                 ;3
            ORI.W     #$8000,SR             ;4
            MOVE.W    D0,D1                 ;5
            ANDI.W    #$F000,D1             ;6
            ANDI.W    #$7FFF,SR             ;7
            MOVE.W    #7,$7EEE              ;8
```

On a line by line basis, indicate what the state of the processor is and any exception processing that will take place.

line	description of what happens
1:	no exception processing, sets the supervisor stack pointer to $8000; in SUPERVISOR mode
2:	exception processing, odd address exception, instruction is NOT executed
3:	exception processing, 1111 unimplemented instruction TRAP
4:	no exception processing, turns TRACE bit ON
5:	instruction does not directly cause an exception; TRACE exception occurs
6:	instruction does not directly cause an exception; TRACE exception occurs
7:	instruction turns the TRACE bit off; TRACE exception occurs
8:	moves $0007 to address $7EEE, no TRACE!!!

10. Explain what is wrong with the following program fragment and correct it. (SR)=$2000

```
LEA       $4000,USP ;cannot access the USP in supervisor
                    mode
ADDX      D0,D1
```

ANSWER: The SR indicates that the 68000 is in supervisor mode. An

LEA cannot have a USP destination; only a MOVEA can have such a destination. Convert to:

```
        LEA     $4000,A0    ;can use any intermediate destination
                            register
        MOVE.L  A0,USP      ;give full credit for MOVEA.L as well
                            IMMEDIATE MODE IS NOT
                            ALLOWED AND IS A WRONG
                            ANSWER.
        ADDX    D0,D1
```

We took off 1 point for a MOVEA; it was a good attempt.

13. Assuming the CPU is in supervisor mode, what does the following code segment do. Be as specific as you can with the information given.

```
LEA       $20000,A0
MOVE.L    A0,USP
MOVE.L    #$10000,-(SP)
CLR.W     -(SP)
RTE
```

Answer: The LEA puts $20000 into A0. The first MOVE.L then sets the value of the USP to the value in A0 ($20000). The second MOVE.L puts the longword $10000 onto the (supervisor) system stack. The CLR basically pushes $0000 onto the system stack. The RTE interprets the 6 bytes on the system stack as a value of the status register and pc and proceeds to pop them off the stack, setting the SR=$0000 (putting the 68000 into user mode) and setting to pc=$10000 starting the program at that memory location. The stack looks like
[$0000] ← the system stack pointer is here before the LEA
[$0001]
[$0000]
[$????] ← the system stack pointer will be here
Scoring: For each instruction -2 points, -4 points for not knowing what program did.

13. A 68000 microcomputer with 1 MByte RAM has the memory contents shown below. Where does the 68000 start executing code AFTER a RESET occurs.

address	contents
$00	$00
$01	$01
$02	$20
$03	$0A
$04	$07
$05	$FF

$06	$10
$07	$E0

ANSWER: The PC is set to the contents of $4, i.e. $07FF10E0, and begins executing the code which begins there.

11. The 68000 is in supervisor mode and executes the following program fragment.

```
MONITOR EQU     $8146               ;starting address of
                                    user program
        MOVE.L  #$3C00,-(SP)        ;load starting address
                                    of $3C00 onto system
                                    stack
        MOVE.W#$8000,-(SP)          ;now load a SR which
                                    is configured for user
                                    mode, trace on,
                                    interrupt level 0
        RTE                         ;pop the SR and PC off
                                    the system stack; start
                                    the program in user
                                    mode at PC=$8146
        JMP     MONITOR             ;execute main user
                                    program
        ORG     $8146               ;jumps here
LOOP    BRA     LOOP                ;infinite loop
```

(a) What do the two MOVE instructions do? A picture of the appropriate stack is expected.

ANSWER: Load the system stack with
SP--->[$8000]
[$0000]
[$3C00]

(b) What does the RTE instruction do?
ANSWER: pops the SR=$8000 off the stack and starts program execution at $$3C00 with the TRACE turned off in <u>user</u> mode.

(c) Will the JMP MONITOR instruction ever be executed? Why or why not?
ANSWER: Probably not since the program jumps off to $0000 3C00.

1. Consider the following program segment: (10 points total)

```
EXCEPT      EQU     $000C
            XREF    STOP

START:      LEA     $8000,SP
NEXT        MOVE.L  #QA,EXCEPT
            LEA     $10001,A1
INST:       JMP     (A1)
            JMP     STOP

QA          ADDA.L  #14,SP
            MOVE.L  #INST+1,-(SP)
            MOVE.W  #$2000,-(SP)
            RTE

            END
```

(a) What does the instruction labeled NEXT do?

ANSWER: Loads the exception vector for an address error with the address of the exception service routine QA.

(b) What is the effect of running the above program?

ANSWER: The LEA instruction causes an odd address error which causes an odd address exception and a subsequent jump to QA. During the course of executing QA a second odd address error occurs with MOVE.L #INST+1,-(SP) and the program causes odd address errors forever.

6. A 68000 microcomputer with 32 MByte RAM has the memory contents shown below. Where does the 68000 start executing code AFTER a RESET occurs.

```
address contents
   $00  [$00]
   $01  [$01]
   $02  [$00]
   $03  [$06]
```

```
$04  [$00]
$05  [$07]
$06  [$10]
$07  [$00]
```

ANSWER: (7 points) The PC is set to the contents of $4, i.e. $00071000, and begins executing the code which begins there.

2. You are executing the following program fragment which begins at $2000 with (SR)=$2700.

```
        ORG         $2000
        MOVEA.L     #$9000,SP
* (c) put instructions to load the exception vector
*     table here
        MOVE.L      #$1700,$2C  ;4 points
        MOVE.L      #Y,-(SP)    ;<--(b)
        MOVE.L      #X,-(SP)    ;<--(b)
        DC.W        $F123       ;<--(a)
        NOP
        MOVE.L      D1,Z
        BRA         EXIT4

        ORG         $1700
EXCPT   NOP

* (d) instructions which retrieve X and Y from stack
DOIT    MOVE.L      6(SP),D0     ;* put X into D0
                                 ;4 points for (e)

        MOVE.L      10(SP),D1    ;* put Y into D1

* (e) instructions to cause return to EXIT4
JUMP    MOVE.L      #EXIT4,2(SP)  ;put EXIT4 on stack
                                  ;4 points

        RTE

        ORG         $4200
Z       DS.L        1
X       EQU         $0100
Y       EQU         $A000


EXIT4   NOP
        END
```

(a) An exception occurs when the 68000 attempts to execute the instruction beginning with the word $F123. Which exception occurs and what is its vector number?

Answer : (4 points total) It is a $1111 exception (2 points) which is

vector number 11 (2 points).

(b) The instructions labeled (b) put X and Y on the stack. Which stack are they put on?
ANSWER: (4 points) the system stack since the program started with SR=$2700 which indicates that the supervisor bit is set.

(c) You want to service this exception with the EXCPT routine beginning at $1700. Place the instructions into the box labeled (c) which will properly load the exception vector table for this to happen.

(d) Assume that you have answered parts (a) and (b) correctly and the 68000 starts to execute your exception service routine beginning at $1700. Place the instructions in the box labeled (d) which will retrieve X and Y from the stack.
NOTES: (1) I want the values of X and Y, not their addresses.
(2) Instructions of the form MOVE.L #X,D0 will receive zero credit.

(e) You want EXCPT to be an exception service routine which returns to EXIT4. Place the instructions in the box labeled (e) which will cause the exception to return, NOT to the "next instruction," but to EXIT4.

10. You have decided that you want to write a routine to elegantly stop your program in the db68k debugger. You write the following program:

```
        ORG       $3000
EXIT5   LEA       EXITMSG,A0    ;load location of
beginning
                                ;of message
        JSR       PutString     ;routine to print
message
        STOP      #$2500        ;processor goes into
HALT
                                ;mode - wakes up only
                                ;for level 5
```

```
interrupts
EXITMSG  DC.B     'PROGRAM EXECUTION BEING TERMINATED.
',0
ENDMSG   EQU      *
```

What do you need to do (give MC68000 code) to make this program execute in response to a TRAP #$F instruction in your main program, i.e. how do you set up the exception vector table to make the 68000 execute this fragment by a TRAP #$F call? Give explicit 68000 code for doing this.

ANSWER: The exception vector number is 32+F=47. According to the rules for exceptions, the exception vector table address is then 47x4=188=$BC. You would then need to put the following instruction (or something equivalent) near the beginning of your program:

```
ANS   MOVEA.L   #$5200,$BC ;load location of
                           ;exception handling routine
```

You lost 5 points if you did not give me an instruction like the above. If you did not include an address you lost four points. You got three points for coming up with the vector number; an additional two points for the exact address, i.e. $BC.

3. The following rather clever program is used to determine the size of memory in a single board 68000 computer system. Explain how it works. (10 points total)

```
VB          EQU       $08
NULL        EQU       $00
            INCLUDE   io.s

            ORG       $11000
* initialize registers and exception vector table

            LEA       MSG,SP            ;SP for exception
                                        ;processing
            MOVE.L    #ENDM,VB

            LEA       ENDPROG,A0

* routine to test memory size by generating a bus
error
SIZE        MOVE.W    #7,(A0)+          ;write data to
memory
            BRA       SIZE              ;until bus error

* bus error exception service routine
ENDM        MOVE.L    A0,D0             ;store first
address
                                        ;past RAM in A0
            LEA       MSG,A0
            JSR       PutString
            SUBQ.L    #8,D0             ;delete storage
for
                                        ;RESET vector
            JSR       HexOut            ;display it

            ORG       $11500
            DS.B      100               ;small stack
MSG         DC.B      'Bytes of available RAM:'null

ENDPROG:    NOP
            END
```

ANSWER: This rather clever program simply writes to memory until it runs our of memory. When it runs out of memory a BUSERROR is generated which causes a jump to the BUSERROR service routine.

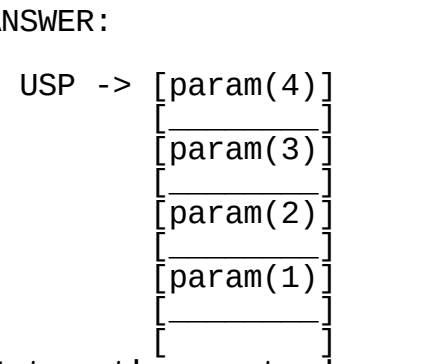
This routine pops the last address of memory off the stack, subtracts 8 bytes for the RESET vector, and then displays the value.

4. The following program is assembled and loaded at address $4000. The programmer calls a TRAP #1 exception which is to access the parameters placed on the stack by his/her program. Assume the program starts in USER mode. (15 points total)

```
                          * THIS IS THE USER'S PROGRAM
                               ORG     $4000
                          START2:
                          * put input parameters on stack
00004000 2F3C 0000 0003        MOVE.L  #3,-(SP)   ;param 1
00004006 2F3C 0000 A500        MOVE.L  #BUF,-(SP) ;param 2
0000400C 2F3C 0000 0200        MOVE.L  #512,-(SP) ;param 3
00004012 2F3C 0000 0002        MOVE.L  #2,-(SP)   ;param 4
00004018 4E41                  TRAP    #1
0000401A 4E71                  NOP     ;end of user program

                               ORG     $4500
00004500                   BUF: DS.B    512
                           * service routine for TRAP #1.
                               ORG     $4700
                           TRAP1:
00004700 48E7 80C0             MOVEM.L D0/A0-A1,-(SP)
                           STACKS:
                           * code for processing TRAP goes here

                           * instructions for part (b) go here
                           *    YOUR ANSWER GOES HERE

0000400A 4CDF 0301             MOVEM.L (SP)+,D0/A0-A1
0000400E 4E73                  RTE
                               END
```

(a) Draw a picture of the system and user stacks at the point labeled STACKS: in the TRAP #1 service routine.

ANSWER:



Note: these stacks are shown as word width instead of the usual byte width for brevity.

(b) Immediately after the label STACKS: in the TRAP #1 service

routine the programmer wants to access the data labeled “param 2” that was put on the stack in the user program and put it into D0. Give 68000 code for doing this.

```
ANSWER:
     MOVEA.L    USP,A1      ;A1 points at user stack
     MOVE.L     8(A1),D0    ;get data with offset
                            ;of 2 long words
```

5. Consider the following program located at $5000. The accompanying interrupt service routine (ISR) beginning at $5500 is for a 68230 driven digital clock similar to your lab #6. You want the 68000 to execute the routine which begins at $5500 in response to any interrupt generated by the 68230. Assume that the 68230's TIVR register has been loaded with the number $80 and that the 68000 is in supervisor mode. (10 points total)

```
            ORG       $5000
ANSWER      ?????                       ;(b)
            MOVE.W    #$2200,SR         ;(a)
LOOP        BRA       LOOP

            ORG       $5500
TIMER:      MOVE.L    A0,-(SP)          ;ISR
            ADDQ.W    #1,TOT_TIME
            LEA       TBASE,A0
            BSET      #0,(TSR,A0)
            MOVE.L    (SP)+,A0
            RTE


TOT_TIME    DS.W      1
TBASE       EQU       $10021
TSR         EQU       20
```

(a) What does instruction (a) do?
ANSWER: The instruction labeled (a) changes the interrupt mask to %010 which disables all external interrupts below level 3. Note that the 68000 MUST be in supervisor mode for you to be able to do this.

(b) What is the function of the TIVR register in the 68230?
ANSWER: It specifies the vector number of the appropriate service routine for the 68230.

(c) The code at $5000 performs several important initialization steps. Put an instruction at (b) which properly loads the Exception Vector Table so that the routine TIMER will be executed in response to these 68230 interrupts.
ANSWER: The routine should be located at 4x$80 = 4x128=512=$200
ANSWER MOVE.L #$5500,$200

9. You execute the following problem. Which exception is caused by this program and why?

```
            INCLUDE io.s            ;include i/o
routines
            ORG     $10000
START       LEA     HWORLD,A0       ;load address of
string
            JSR     STRLENGTH       ;calculate string
length
            JSR     PutString       ;output it
            NOP                     ;rest of your
program

            ORG     $10200          ;subroutine
STRLENGTH   MOVE.L  A0,-(SP)        ;put address on
stack
            MOVEQ   #0,D0           ;initialize counter
LOOP        TST.B   (A0)+           ;is byte $0?
            BEQ     OUT             ;if yes then finish
            ADDQ.W  #1,D0           ;else increment
counter
            BRA     LOOP            ;and continue
looping
OUT:        RTS                     ;get out of here

HWORLD:     DC.B    'Hello world!',0
            END
```

ANSWER:
You put the address of the string on the stack above the return address and never pop it off the stack before the return. As a consequence the program returns to the address of HWORLD and attempts to execute a string. This will typically cause an illegal instruction exception since "He" does not correspond to a valid instruction.

RISC / ARCHITECTURE

7. Indicate whether the following statements are true or false. (1 point per answer)

	True	Fals e
All computers which contain a pipeline are characterized as RISC.		F
RISC processors contain a few specialized registers.		F
RISC processors are faster than CISC processors.		F
RISC processors use only one data type, i.e. words.		F
All RISC instructions are the same length.	T	
RISC processors have many instructions for accessing and manipulating memory.		F

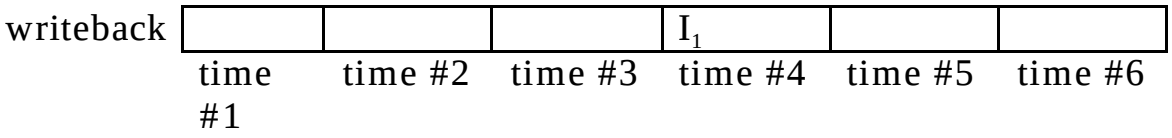
8. Consider a 4 stage pipelined processor which has processing units for fetch,decode,execute, and writeback. NOTE: A writeback means that the results are then written to a register.

(a) Sketch the temporal execution of a typical instruction sequence.

	time #1	time #2	time #3	time #4	time #5	time #6
fetch	I_1	I_2	I_3	I_4	I_5	I_6
decode		I_1	I_2	I_3	I_4	I_5
execute			I_1	I_2	I_3	I_4
writeback				I_1	I_2	I_3

(b) Sketch the corresponding operation of a conditional branch instruction and explain why such instructions degrade pipeline performance.

fetch	I_1				I_2	I_3
decode		I_1				I_2
execute			I_1			



(c) Discuss the following statements.

(i) All branch statements cause pipeline "bubbles."

ANSWER: No, all unconditional jumps are fine. Only conditional branches cause bubbles.

(ii) The Decrement and Branch form of branch is "well suited" for pipeline processors.

ANSWER: Since the Decrement and Branch instruction always branches (except for the last time when it quits and falls through) it only creates one bubble and is highly efficient when pipeline processed.

10. What is the output of the following program? Does it ever terminate? Assume that the program starts in supervisor mode?

```
V1010     EQU     $28
VTRAP0    EQU     $80
VTRACE    EQU     $24

START     LEA     $8000,SP          ;1-set the SSP
          MOVE.L  #EMU,V1010        ;2-%1010 instruction trap
          MOVE.L  #PR,VTRAP0        ;3-TRAP #0
          MOVE.L  #TRACE,VTRACE     ;4-trace exception

          MOVEQ   #0,D7             ;5-put 0 into D7
          JSR     INIT_PORT         ;6-initialize 6850
          DC.W    $A000             ;7-force 1010 exception
          ORI.W   #$8000,SR         ;18-RTE returns here
                                    ;19-turns trace on
          MOVE.W  #1,D0             ;20-do meaningless moves
                                    ;21-trace prints 1
          MOVE.W  #2,D0             ;22-trace prints 2
          MOVE.W  #3,D0             ;23-trace prints 3
          ANDI.W  #$7FFF,SR         ;24-this is a priviledged
                                    ;instruction - turns
                                    ;off trace
          JSR     STOP              ;25-stops program

ID        BRA     ID

PROGCNT   EQU     6

EMU       LINK    A6,#0             ;8-ISR stack frame
          MOVEM.L D0/D7/A0,-(SP)    ;9-now save registers
          MOVEQ   #7,D7             ;10-set D7=7
          MOVE.L  (PROGCNT,A6),A0   ;11-put RA into A0
          MOVE.W  (A0),D0           ;12-get offending $A000
          JSR     HexOut            ;13-print it out
          ADDI.L  #2,(PROGCNT,A6)   ;14-add 2 to RA on stack
          MOVEM.L (SP)+,D7/D0/A0    ;15-restore registers
          UNLK    A6                ;16-unlink
          RTE                       ;17-pop SR & PC

TRACE     TRAP    #0                ;calls TRAP0
          RTE

PR        MOVE.L  D7,-(SP)          ;save D7 on stack
```

```
          JSR     HexOut          ;print out D0
          MOVE.L  (SP)+,D7        ;restore D7
          RTE

          END
```

NOTE:
HEXOUT Prints to the debugger screen the hex word in D0.

ANSWER: This program executes the 27 instructions shown in sequence. The printout is

$A000	will actually print	$FFFFA001
$0001		$00000001
$0002		$00000002
$0003		$00000003

<program stops>

Either column would be perfectly acceptable.

-3 points, not realizing that DC.W at 7 forces a 1010 exception and goes to EMU
-2 points, not getting the print of $A000 in emu
-2 points, returns to 18/19 and turns TRACE on
-2 points, lines 20/23 print $0001, $0002, and $0003
-1 point, lines 24 and 25 turn off trace and stop program

FOR YOUR REFERENCE THE EXCEPTION VECTOR TABLE IS:

vector number (Decimal)	address (Hex)	assignment
0	0000	RESET: initial supervisor stack pointer (SSP)
1	0004	RESET: initial program counter (PC)
2	0008	bus error
3	000C	address error
4	0010	illegal instruction
5	0014	zero divide
6	0018	CHK instruction
7	001C	TRAPV instruction
8	0020	priviledge violation
9	0024	trace
10	0028	1010 instruction trap
11	002C	1111 instruction trap
12*	0030	not assigned, reserved by Motorola
13*	0034	not assigned, reserved by Motorola
14*	0038	not assigned, reserved by Motorola
15	003C	uninitialized interrupt vector
16-23*	0040-005F	not assigned, reserved by Motorola
24	0060	spurious interrupt
25	0064	Level 1 interrupt autovector
26	0068	Level 2 interrupt autovector
27	006C	Level 3 interrupt autovector
28	0070	Level 4 interrupt autovector
29	0074	Level 5 interrupt autovector
30	0078	Level 6 interrupt autovector
31	007C	Level 7 interrupt autovector
32-47	0080-00BF	TRAP instruction vectors**
48-63	00C0-00FF	not assigned, reserved by Motorola
64-255	0100-03FF	user interrupt vectors

NOTES:

* No peripheral devices should be assigned these numbers
** TRAP #N uses vector number 32+N

The following logic functions may be needed at various points throughout the exam.

A	B	A OR B	A AND B	A EOR B
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0