1. bit manipulation instructions such as BTST, BCHG

9. (Exam #2, 1990)
If ($53EFE)=$ 0A ($5FFFE)=$ 00
($53EFF)=$ EE ($5FFFF)=$ 00
($53F00)=$ FF ($60000)=$ 00
($53F01)=$ 00 ($60001)=$ 00
($53F02)=$ 12 ($60002)=$ 00
($53F03)=$ 3F ($60003)=$ 00
($53F04)=$ 30 ($60004)=$ 00
and
SR=$ 05 00
what is the result of the instruction BCHG #$0C,$53F00?

ANSWER:
There is a trick to this problen, the BCHG argument cannot be larger than 8 when used on memory so the bit change is $C MOD
8 or 4. This changes the byte located at $53F00 to EF, i.e.
1111 1111 becomes 1110 1111 and ($53F00)=$EF.

8. (Exam #2, 1991) If (D1.L)=$ABCD 1A97 and (SR)=$ 0500, what is the value of SR and D0 after the following code is executed?

```
MOVEQ     #5,D0
BTST D0,D1
```

(D1.L) = _____
(SR) = _____

ANSWER:
(D1.L) = $ABCD1A97
(SR) = $0504
The BTST tests bit 5 (actually the sixth bit) of D1 and places the NOT of that bit in the Z bit of the SR. In this case bit 5=0 and, and a result, Z=1

2. labels and the symbol table

11. (Exam #1, 1992)
Assume (D0)=$ 0005 3F02 and (D1) = $ FFFF 0110

```
        ORG      $1000
        MOVE.W   #A,D0      (a)
        MOVE.B   CNT,D1     (b)

        ORG      $2000
CNT     DC.W     $1
BLANK   DC.L     $FFA0
SUM     DS.W     1
A       EQU      $17
ICNT    DC.W     3
```

(a) What is in memory beginning at memory location $2000. Any memory contents not explicitly stated by the code may be
indicated by "xx," i.e. if the program doesn't put anything in that location its contents are "xx."

($2000) =
($2001) =
($2002) =
($2003) =
($2004) =
($2005) =
($2006) =
($2007) =
($2008) =
($2009) =
($200A) =
($200B) =
($200C) =
($200D) =

(b) What is in D1 after the instruction labeled (b) is executed?

(D1) = _____

ANSWERS:
(a)
($2000) = $00
($2001) = $01
($2002) = $00
($2003) = $00
($2004) = $FF
($2005) = $A0
($2006) = $xx
($2007) = $xx
($2008) = $00
($2009) = $03
($200A) = $xx
($200B) = $xx
($200C) = $xx
($200D) = $xx

(b) (D1) = $FFFF0100

10. (Exam #1, 1990)
Given the program segment

```
           ORG      $7000
*          TABLE OF NUMBERS
FTABLE     DC       7000
           DC       9040
           DC       120
           DC       720
VALUE      DS.B     1
           DS.B     1
RESULT     DS.W     1
           ORG      $4000
           CLR.L    D0
           CLR.L    D1
           MOVEA.L  #0,A0
           MOVEA.L  #0,A1
           MOVEA    FTABLE, A0
           ADDA     #FTABLE, A0
           END
```

What is in A0 AFTER I execute the program beginning at $4000?
(A0) = _____

ANSWER: The two CLR's have nothing to do with address registers. The first MOVEA.L basically clears A0. The second one does the same to A1 (not really relevant to the question). The MOVEA FTABLE,A0 puts the word length contents of the address (which is $7000) into A0 sign-extending this number as necessary, i.e. (A0)= ($0000 7000). Since this number is not given we cannot give a specific answer.

The ADDA #FTABLE works in a similar manner. #FTABLE is equivalent to #$7000. The number $7000 will be sign-extended as appropriate. Since the leading bit of $7000 is a zero, only zeros will be put at the beginning of the number. The calculation then adds $0000 7000 to $0000 7000, placing the long word result $0000 D000 into A0.

9. (Exam #1 1991) Assume (D0)=$ 0005 3F02, (D1) = $ FFFF 0110

```
        ORG      $1000
        ADD.L    #DATA,D0      (a)
        MOVE.W   DATA,D1       (b)

        ORG      $2000
DATA    DC.B     $0A, $EE, $83, $82
        DC.B     $0A, $EE, $30, $00
```

(a) What is in D0 after the instruction labeled (a) are executed?
(D0) = _____

(b) What is in D1 after the instruction labeled (b) are executed?
(D1) = _____

ANSWERS:
(a)
(D0) = $00 05 5F 02

(b)
(D1) = $FF FF 0A EE

3. debugger and UNIX commands

2. Where are each of the following commands used (vi, UNIX, debugger, etc) and describe their function (a sentence or two):

		where	function
a)	dd	vi	delete current line
b)	:wq	vi	save file and quit
c)	ls	UNIX	display current directory contents
d)	cd ..		UNIX change directory to parent directory

2. Where are each of the following commands used (vi, UNIX, debugger, etc) and describe their function (a sentence or two):

		where	function
a)	lp	UNIX	direct the following file to the printer
b)	:q!	vi	in command mode, exit without saving
c)	man	UNIX	display documentation about following subject
d)	x	vi	delete character under cursor

3. Answer the following questions about the debugger screen shown below:

```
=======================Monitor====================12===========Stack=======14==
|1                                                  ||    00000010=4E724E72
|
|2                                                        0000000C=4E724E72
|
|3                                                        00000008=4E724E72
|
|4                                                        00000004=4E724E72
|
|5                                                      SP->00000000=4E724E72
|
================================================================================
                        Code                      11
=======Registers=====13==
  00004002 307C 6000         MOVEA.W #$6000,A0         |PC=0000400A
pi=00004006|
  00004006 3278 6000         MOVEA.W DATA,A1            D0=00000000
A0=00006000|
  0000400A 347C 6000         MOVEA.W #$6000,A2          D1=00000000
A1=00000000|
  0000400E 303C 6000         MOVE.W  #$6000,D0          D2=00000000
A2=00000000|
  00004012 3238 6000         MOVE.W  DATA,D1            D3=00000000
A3=00000000|
```

```
  00004016 3628 0006         MOVE.W  $6(A0),D3          D4=00000000
A4=00000000|
  0000401A 31FC 0005 6010    MOVE.W  #$5,VALUE          D5=00000000
A5=00000000|
 fact:                                                 D6=00000000
A6=00000000|
  00004020 3A38 6010         MOVE.W  VALUE,D5          |D7=00000000
A7=00000000|
  00004024 DA45              ADD.W   D5,D5              SR=0010011100000000
|
| 00004026 47F8 6000         LEA     DATA,A3               T S  III   XNZVC
|

=========================
STATUS: Command         68000   MODULE: labels_2          BREAK #: 0   HELP=F5
> Program Step
  Breakpoint  Debugger  Expression  File  Memory  Program  Symbol  Window
  Display_Source  Find_Source  Run  Interrupt  Load  Context  Pc_Reset  Step
```

a) The NEXT 68000 instruction the debugger will execute is:
ANSWER: the debugger is at $400A, MOVEA.W #$6000,A2

b) How could you make the debugger execute the instruction at
$4020 NEXT? Please give specific debugger command(s) you would use.
ANSWER: set the PC to $4020 by Memory Register @PC=$4020
Other answers such as Program Step From $4020 or Program Run
From $4020 are also correct.

3. Where are each of the following commands used
(vi, debugger, UNIX, etc) and describe their function
(a sentence or two):

(a) ESC vi exit text entry mode
(b) :wq vi go to command mode, write the
current buffer to a file (assigned
when vi was invoked), and exit vi
(c) cd UNIX return to the user's home directory
(d) news -a UNIX re-read all previously read

4. Answer the following questions about the debugger screen
shown below:

```
======================Monitor====================12===========Stack=======14=
=
|1                                                ||    0o000010=4E724E72
|
|2                                                      00o0000C=4E724E72
|
|3                                                      00000008=4E724E72
|
|4                                                      00000004=4E724E72
|
|5                                                    SP->00000000=4E724E72
|
================================================================================
=
                        Code                      11
=======Registers=====13==
  00004002 307C 6000         MOVEA.W #$6000,A0         |PC=0000400E
pi=0000400A|
  00004006 3278 6000         MOVEA.W DATA,A1            D0=00000000
A0=00006000|
  0000400A 347C 6000         MOVEA.W #$6000,A2          D1=00000000
A1=00000000|
  0000400E 303C 6000         MOVE.W  #$F000,D0          D2=00000000
A2=00000000|
  00004012 3238 6000         MOVE.W  DATA,D1            D3=00000000
A3=00000000|
  00004016 3628 0006         MOVE.W  $6(A0),D3          D4=00000000
A4=00000000|
```

```
  0000401A 31FC 0005 6010    MOVE.W  #$5,VALUE          D5=00000000 A5=00000000
 fact:                                                  D6=00000000
A6=00000000|
  00004020 3A38 6010         MOVE.W  VALUE,D5          |D7=00000000
A7=00000000|
  00004024 DA45              ADD.W   D5,D5              SR=0010011100000000
|
| 00004026 47F8 6000         LEA     DATA,A3               T S  III   XNZVC
|

=========================
STATUS: Command         68000   MODULE: labels_2          BREAK #: 0   HELP=F5
> Program Step
  Breakpoint  Debugger  Expression  File  Memory  Program  Symbol  window
  Display_Source  Find_Source  Run  Interrupt  Load  Context  Pc_Reset  Step
```

(a) The NEXT 68000 instruction the debugger will execute is:

ANS: The debugger PC is at $400E causing execution of the instruction MOVE.W #$F000,D0

(b) If you execute the debugger command shown, describe any changes that will occur to the values shown in the Registers window? (You may ignore the pi value.)
ANS: The next instruction will be executed, the pc will advance to $0000 4012, and $F000 will appear in D0. Also, the values of the Status Register will change to:
0010011100001000
T S III XNZVC
i.e., the N bit is set.

Students who answered the question using the machine code 303C 6000 rather than the mnenomic were eligible for partial credit.

4. address register indirect addressing modes

6. (exam #2, 1991)
If ($53EFE)=$ 0A ($53F05)=$ 00
($53EFF)=$ EE ($53F06)=$ FF
($53F00)=$ 03 ($53F07)=$ AA
($53F01)=$ 82 ($53F08)=$ EE
($53F02)=$ 0A ($53F09)=$ AB
($53F03)=$ EE ($53F0A)=$ 02
($53F04)=$ 30 ($53F0B)=$ 82
and
(A0)=$ 0005 3F08
what is A0 and what memory, if any, is changed after the instruction
MOVE.L -(A0), -(A0)
is executed?

(A0.L) = _____

ANSWERS

The memory contents change to
($53EFE)=$ 0A ($53F05)=$ 00
($53EFF)=$ EE ($53F06)=$ FF
($53F00)=$ 03 -> $30 ($53F07)=$ AA
($53F01)=$ 82 -> $00 ($53F08)=$ EE
($53F02)=$ 0A -> $FF ($53F09)=$ AB
($53F03)=$ EE -> $AA ($53F0A)=$ 02
($53F04)=$ 30 ($53F0B)=$ 82
and
(A0)=$ 0005 3F00

Evaluated from left to right, the address is decremented by 4
(it's a long word), to $3F04. The instruction fetches ($3F04.L)=$3000FFAA, computes the destination address as $00053F04-00000004=$00053F00, and puts the long word $3000FFAA there. Note that this instruction does not put the
contents of memory into A0.

5. address register manipulation instructions such as LEA, ADDA, MOVEA

5. (Exam #2 1992) What is in D2 and A0 after executing the code beginning at START?

```
         ORG     $6000
         DC.W    $A20B
         DC.W    $0000

LENGTH   EQU     $6000
A        EQU     $A001
B        EQU     $6101

START    CLR.L    D2          MOVE.W   LENGTH,D2
         MOVEA.L  #A,A0
         LEA      0(A0,D2.W),A0
(D2)=_____________
(A0)=_____________
```

ANSWER:
(D2)= $0000A20B
(A0)= $0000420C

The key point is what the instructions do. Beginning at START the instructions do the following:

```
START    CLR.L    D2             ;this instruction
                                 ;clears D2
         MOVE.W   LENGTH,D2      ;this instruction
                                 ;moves $A20B to D2
         MOVEA.L  #A,A0          ;this instruction
                                 ;moves $0000A001
                                 ;to A0
         LEA      0(A0,D2.W),A0  ;this instruction
                                 ;adds $0000A001 to
                                 ;$FFFFA20B to get
                                 ;$0000420C
```

6.

If ($53EFE)=$ 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

and

(A0) = $ 0005 3F08,
(A2) = $ 0000 0000,
(D0) = $ FFFF 0002 and
(D1) = $ 0000 0002

before each instruction is executed, what is the result of each of the following instructions?

```
(a)     LEA      $53F08,A0
(b)     MOVEA.W  (A0),A2
(c)     MOVE     (A0)+,D1
(d)     MOVE     $FE(A0,D0.W),D0
```

ANSWERS:

```
(a)     LEA      $53F08,A0
```

RESULT: (A0) = $0005 3F08
The instruction puts $53F08 into A0. It DOES NOT put the contents of $53F08 into A0.

```
(b)     MOVEA.W  (A0),A2
```

RESULT: (A2) = $FFFF EEAB
The instruction fetches $EEAB. It then sign extends it to $FFFF EEAB since it is a MOVEA.W and puts it into A2.

```
(c)     MOVE     (A0)+,D1
```

RESULTS: (D1) = $0000 EEAB, (A0)=$0005 3F0A
The instruction fetches the word contents of $53F08, i.e. $EEAB and puts this into D1. $EEAB is not sign extended. A0
is incremented by two bytes to $53F0A.

(d) MOVE $FE(A0,D0.W),D0
ANSWER: (D0)=$FFFF 0282
This is a tricky problem. The address performs the addition
of three numbers
$ 0005 3F08 A0
$ 0000 0004 D0.W sign-extended to long word (+2 decimal)
$ FFFF FFFE $FE sign-extended to long-word (-2 decimal)
$ 0000 3F0A this is the result
It fetches $0282 and puts it in the lower word of D0. The source address calculation doesn't alter the contents of D0.

5. (Exam #2, 1991)
Assume that (A1) = $ 0000 1000, (A2) = $0000 8002, (D1) = $
FFFF 0000, (D2)= $0000 0000, and (D3)= $0000 0000. Indicate
what is in D3, A3 and A4 after the following program fragment
is executed:

```
      ORG      $5000
A:    DC.B     $10,$02,$30,$13,$B0,$21,$4A,$31
B:    DC.L     $C1D21122
      MOVE.L   #$FFFC,D3
      LEA      B,A3
      LEA      4(A3,D3.W),A4
      MOVEA.W  -4(A4),A4
```

Determine:
(D3.L) = _____
(A3.L) = _____
(A4.L) = _____

ANSWERS:
(D3.L) = $ 0000 FFFC
(A3.L) = $ 0000 5008
(A4.L) = $ FFFF B021 by sign extension.

EXPLANATION: The MOVE.L instruction produces (D3.L)=$0000FFFC. The LEA B,A3 instruction produces (A3.L)=$00005008 since B is located 8 bytes from A which began at $00005000. The second LEA instruction computes the
address as A3+D3.L+4 = 00005008 + FFFFFFFC + 00000004 =

00005008 where the contents of D3 were sign extended. Since FFFFFFFC is -4, this exactly canceled out the +4 offset and the address remained $00005008. This address are then placed into A4. Many students incorrectly put $C1D11122 into A3. An LEA never puts the contents of an address into an address register; it computes the address and puts the address into the data register. The final instruction MOVEA.L subtracts 4 from $00005008 to get $00005004. The instruction fetches the word contents of that address ($B021) and sign extends to put $FFFFB021 into A4.

3. (Exam #2, 1991) Assume the following program segment is executed with (D0)=$FF0100AA initially.

```
                              ORG     $6000
006000 203C FF01 00AA   MAIN  MOVE.L #$FF0100AA,D0
006006 203C 0000 FF06         MOVE.L  #$FF06,D0
00600C 0240 00FF              ANDI.W  #$00FF,D0
006010 41F8 601A 4E71         LEA     AMP,A0
006016 4EF0 0000         J:   JMP     0(A0,D0.W)

00601A E848              AMP: LSR     #4,D0
00601C 0640 0012              ADD     #$12,D0
006020 0440 000A              SUB     #$0A,D0
006024 E548                   LSL     #2,D0
006026 4E71              E:   NOP
```

COMMENT: Ignore everything after the label AMP:

(a) Before the instruction labeled J is executed what are the following values:
(D0.L) = _____
(A0.L) = _____

(b) What does the instruction at the label J do?

COMMENT: Ignore what the instruction JMP does and pay attention to how the address is computed.

ANSWER:

(a)
(D0.L) = $0000 0006
(A0.L) = $0000 601A

(b) The first MOVE is basically superseded by the second MOVE
instruction which put $0000FF06 into D0. The ANDI with $00FF
changes the upper byte of the lower word in D0 to $00 with
the result that (D0.L)=$00000006. The LEA instruction loads
the address of AMP ($0000601A) into A0. The JMP instruction
computes the address 0000601A+00000006 = 0000 6020 and jumps
to it. In this case, the JMP goes to the SUB instruction and
continues program

6. simple conditional loops

If (A0)=$0000 1000 and (D0)=$EEAB 0282 what does the following code do? An explanation of what each line does would improve your chances of partial credit.

```
address   memory
     $00
     $01
$1000 --> $22
     $40
     $3D
     $17
     $B0

R       EQU      2
S       EQU      3
RX      EQU      4

IPOLL   BTST     #RX,S(A0)
        BEQ      IPOLL
        MOVE.B   R(A0),D0
(A0.L) = _____________________(D0.L) = _____________________
```

Answer:
(A0.L) = $0000 1000
(D0.L) = $EEAB 023D

The BTST instruction is equivalent to BTST #4,3(A0) which adds 3 to $00001000 to generate the address $00001003. It then tests bit #4 (actually the fifth bit since bits start at
zero) of ($1003).B = $00010111. This bit is one so the BEQ is never executed and program control goes to the next instruction MOVE.B R(A0),D0 which is equivalent to MOVE.B 2(A0),D0. This instruction then moves the contents of the byte at $00001002, i.e. $3D, to D0 so that (D0)=$EEAB 023D.

This was a hard problem to grade. Some guidelines depending
upon the student explanations were:
-4 if branching was wrong
-2 for bit in D0.L being off (usually by 1)
-2 for addressing errors, usually numbering memory by words rather than bytes as shown
-1 if answer was numerically wrong, otherwise ok

3. (Exam #2, 1990) For the following program segment:

```
     CLR.L     D1
     MOVE.L    #10,D0
LOOP:     ADD.L     D0,D1
     SUBQ #1,D0
     BPL  LOOP
     <next instruction>
```

SUBQ gets executed _____times and (D1) = _____
after the program stops.

ANSWER: SUBQ gets executed 11 ($B) times and (D1)=55 ($37)
after the program stops.

7. shift and rotate instructions

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
```

ANSWERS:
(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
```

ANSWER:
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
```

8. polled i/o

2. (Exam #2, 1991) Using the input/output routines introduced in Lab #1, complete the following code fragment to
print out the character "x".

```
     MOVE        ______ , D0
     JSR         ________
```

ANSWER:

```
     MOVE.W      #'x',D0
     JSR         CharOut
```

Note that the character MUST be preceded by a # sign for this
code to work.

2. (Exam #2, 1992) You used polled i/o in Lab #3. Identify
each of the following statements about polled i/o as "True" or "False."

True	False	Statement
T		In polled i/o, the CPU is in a loop waiting for input.
	F	In polled i/o, the CPU can do other things while it is waiting for an input.
	F	Polling only works on 68000's
T		Polling is inefficient compared to other methods of program input.

9. signed and advanced branch instructions

4. (Exam #2, 1990) Let (D0)=$000067C5, (D1)=$0000D29E, and (D2)=$00000000. If the instruction
CMP.W D1,D0
is executed

(a) Assuming the status register is initially,
X N Z V C
1 1 0 1 1
what is in the status register after the CMP is executed?

ANSWER:
X N Z V C
1 1 0 1 1

Note that this computes $000067C5 - $0000D29E = $00009527. This is a positive number minus a negative number which is equivalent to a positive number plus a positive number and the result is negative.

(b) Indicate whether the following branches to <label> will take place after the CMP is executed:
BCS <label> [ ] will branch [ ] will not branch
ANSWER: Will branch since C=1

DBNE D2,<label> [ ] will branch [ ] will not branch
ANSWER: Will not branch since Z=0. Will simply fall through to next instruction.

BPL <label> [ ] will branch [ ] will not branch
ANSWER: Will not branch since N=1

DBPL D2,<label> [ ] will branch [ ] will not branch
ANSWER: Will not branch. Tough. N=1 so PL (positive) is false, therefore D2 is decremented by 1 to -1, so it does not
branch

BGT <label> [ ] will branch [ ] will not branch
ANSWER: Will branch. TOUGH. Logic for GT:
NV~Z+~N~V~Z=1•1•1+0•0•1=1+0=1 Therefore GT is true so it

branches

4. (Exam #2, 1991) Assume that (D0.W) = $E48F and the instruction CMPI.W #7002,D0
is executed.

(a) What is in the status register after the CMP is executed?

X	N	Z	V	C

Answer:

X	N	Z	V	C
-	1	0	0	0

Note that this computes $E48F - 7002 .
The $E48F is a negative number, the number 7002 is a positive
decimal number. ($1B5A) . We have a negative number minus a
negative number giving a negative number. No overflow!

(b) Indicate whether the following branches to <label> will take place after the CMP is executed:

BVC <label> [ ] will branch [ ] will not branch
ANSWER: Will branch since V=0

BGT <label> [ ] will branch [ ] will not branch
ANSWER: Will not branch since

NOTE: These answers are based upon the actual CCR from (a). If you got part (a) wrong, your answers might be different.

If you ignored the fact that 7002 is decimal and treated it as hex you will get the following

X	N	Z	V	C
-	0	0	1	0

In this case, neither instruction will branch. Several other
answers were commonly gotten by ignoring the fact that 7002 is negative or by reversing the order of subtraction, or both. The answers for these cases are:

computed	XNZVC	BVC	BGT
$E48F-$7002	-0010	No	No
$7002-$E48F	-1011	No	Yes
7002-$E48F	-0001	Yes	No

The correct interpretation was

$E48F-7002 -1000 Yes No
where - indicates that the X bit did not change.

4. (Exam #2, 1992) Assume that (D0) = $F0DDE48F, (D1)=$01ABD29E and the instruction
CMP.L D0,D1
is executed.

(a) What is in the status register after the CMP is executed?
X N Z V C

Answer:
This one is very tricky as it computes $01ABD29E - $F0DDE48F
which is a subtracting a negative number, i.e. adding a positive number. The result is $10CDEE0F. The resulting SR
is then

X N Z V C
- 0 0 0 1

The result is not zero. A positive - (negative) is equivalent to positive+positive. No overflow occurs. The result is not negative. However, a borrow (a carry occurs) so C=1. The CMP instruction does not change the X bit.

(b) Indicate whether the following branches to <label> will take place after the CMP is executed:
BCC <label> [ ] will branch [ ] will not branch

ANSWER: Will not branch since C=1

BNE <label> [ ] will branch [ ] will not branch

ANSWER: BNE will branch since Z=0
NOTE: These answers are based upon the actual CCR from (a). If you got part (a) wrong, your answers might be different.

(graded -2 per bit of SR, -2 per branch, -1 for hex interpretation)

10. DBcc insructions

3. (Exam #2, 1992) If (D3)=$FFFF0001, (D4.W)=$0000 0006, determine the value of D3 and D4 after the following code is executed:

```
LOOP:     ...
          CMPI.W    #3,D4
          DBGT D3,LOOP
ENDING:
```

(D3) = _____

(D4) = _____

ANSWER: The CMPI will compute the difference $0006 - $0003 which is $0003. The resulting SR bits will be (XNZVC)=-0000.
The DBGT will fall through to ENDING because destination>source and the program will not change either register.

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

```
        ORG     $4000
        MOVE.B  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

PROGRAMS WHICH ENCOMPASS SEVERAL OF THE ABOVE AREAS:

1. What does this program do? Be as specific as you can.

```
1                                       XREF    HEXIN_LONG
2                                       ORG     $1000
3  00001000 41F8 1014 4E71 START        LEA     I,A0
4  00001006 72F8                        MOVE.L  #-8,D1
5  00001008 D1C1                        ADDA.L  D1,A0
6  0000100A 4EB9 0000 0000              JSR     HEXIN_LONG
7  00001010 2080                        MOVE.L  D0,(A0)
8  00001012 4E40                        TRAP    #0
9
10 00001014                     I:      DS.W    10
11                                      END

Symbol Table

Label           Value

HEXIN_LONG    External
I             00001014
START         00001000
```

Explanation:

The LEA puts the address $00001014 into A0. The first move puts a -8 (decimal) into D1. The ADDA computes the new address $00001014 - $00000008 = $0000100C. The JSR HEXIN_LONG puts a new number into D0. The MOVE.L D0,(A0) puts that new number into the address field of the JSR instruction.

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
       MOVE.W   #10,D2
       MOVEQ    #0,D0
       ANDI.L   #$0000FFFF,D1

ONE:   LSR.W    #1,D2
       BCC.S    TWO

       ADD.L    D1,D0
TWO:   LSL.L    #1,D1

       TST.W    D2
       BNE.S    ONE
       TRAP     #0
```

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

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

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)

```
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
```

```
(D0.L) = $ 00000082  (this is the product)
(D1.L) = $ 000000D0 (this was not altered by the program)
(D2.L) = $ FFFF0000 (it counted down)
```

9. Consider the following program segment:
(COMMENT: This program uses an ADDX which will not be on the
exam, but the rest of the program is reasonable)

```
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) 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 probably not be
on the exam, but the rest of the problem is quite 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

8. (Exam #2, 1992) If (A0)=$00001000 and (D0)=$EEAB0282 what does the following code do? An explanation of what each
line does would improve your chances of partial credit.

```
address    memory
            $00
            $01
   $1000--> $22
            $40
            $3D
            $17
            $B0


R          EQU     2
S          EQU     3
RX         EQU     4

IPOLL      BTST    #RX,S(A0)
           BEQ     IPOLL
           MOVE.B  R(A0),D0
(A0.L) = _____________________
(D0.L) = _____________________
```

Explanation:
(A0.L) = $00001000
(D0.L) = $EEAB023D
Answer: The BTST instruction is equivalent to BTST #4,3(A0) which adds 3 to $00001000 to generate the address $00001003. It then tests bit #4 (actually the fifth bit since bits start at zero) of ($1003).B = $00010111. This bit
is one so the BEQ is never executed and program control goes
to the next instruction MOVE.B R(A0),D0 which is equivalent to MOVE.B 2(A0),D0. This instruction then moves the contents of the byte at $00001002, i.e. $3D, to D0 so that (D0)=$EEAB 023D.

This was a hard problem to grade. Some guidelines were:
-4 if branching was wrong
-2 for bit in D0.L being off (usually by 1)
-2 for addressing errors, usually numbering memory by words rather than bytes as shown
-1 if answer was numerically wrong, otherwise ok