AUGUSTANA UNIVERSITY COLLEGE

CSC 250 — COMPUTER ARCHITECTURE I

Final Exam — 200x December xx

NAME: _____ Instructor: J. Mohr

Time allowed: 3 hours. Closed book - no references allowed except for the tables of Pep/7 opcodes, ASCII codes and other data provided. You may use a calculator.

Questions 1 to 12 are to be answered in the space provided beneath each question. The remaining questions are to be answered on the sheets of lined paper provided. Indicate clearly the question number to which each response corresponds, and label continuations clearly.

The point value of each question is indicated in square brackets in the left-hand margin.
Total points = 110.

---

[4] For questions 1 to 4, circle the letter preceding the best response to each question.

1. Which of the following responses lists the phases of the von Neumann execution cycle in the correct order?
   a. decode, fetch, execute, increment
   b. fetch, decode, execute, increment
   c. fetch, decode, increment, execute
   d. increment, fetch, decode, execute

2. What is the primary advantage of immediate addressing over direct addressing?
   a. The instruction executes faster.
   b. The operand can be a symbol.
   c. It sets more status codes.
   d. It is more easily translated by an assembler.

3. The value of an .EQUATE symbol is like a(n)
   a. address. b. character.
   c. constant. d. variable.

4. Which of the following instructions cannot produce an overflow?
   a. ASLA b. ASRA
   c. COMPA d. DECI

[8] 5. Assuming that the following 8-digit hex numbers represent values in IEEE 754 single-precision floating point, write each value as a signed binary fraction with a base-2 exponent (e.g., $1.1011_2 \times 2^4$) *or*, where applicable, as one of the special values representable in the IEEE 754 format.

   a. F2B40000 _____ c. FFC10001 _____

   b. 7F800000 _____ d. 00340000 _____

[8] 6. Indicate the **range** of numbers that can be represented by each of the following binary representations by specifying the lowest and highest representable numbers in both binary and decimal.

a. unsigned binary in a 5-bit cell

binary – low: _____ high: _____

decimal – low: _____ high: _____

b. signed binary using sign-magnitude representation in a 7-bit cell

binary – low: _____ high: _____

decimal – low: _____ high: _____

c. signed binary using two's complement representation in a 6-bit cell

binary – low: _____ high: _____

decimal – low: _____ high: _____

d. signed binary using excess-7 representation in a 4-bit cell

binary – low: _____ high: _____

decimal – low: _____ high: _____

[12] 7. Assuming **seven-bit two's complement** binary representation, show the results of the following operations, including the effect on the carry and overflow bits where indicated.

a) `000 1111`
AND `101 0110`

C = V =

b) `001 1100`
OR `101 0010`

C = V =

c) **ASL** `010 0101`

C = V =

d) `000 1101`
ADD `011 0100`

C = V =

e) `111 1001`
ADD `000 1101`

C = V =

f) **ASR** `110 0011`

C = V =

g) **NEG** `0101100`

h) **NOT** `0101100`

[2] 8. What output is produced by the following assembly language program?
(Warning: this is a nonsense program that exploits the nature of a von Neumann machine.)

```
start:    HEXO start,d
          STOP
          .END
```

Output:

[12] 9. Show the hex digits that would be inserted into a Pep/7 machine-language program when each of the following assembly language statements is assembled. If no data bits are emitted for the assembly language statement, write "none".

a. `BRLE h#00A7,i` _____

b. `.BLOCK d#4` _____

c. `ASLX` _____

d. `linefd: .EQUATE h#000A` _____

e. `STBYTA ,x` _____

f. `RTS` _____

g. `num: .WORD d#12` _____

h. `LOADA d#-2,s` _____

i. `msg: .ASCII /Hi/` _____

j. `DECI h#0032,d` _____

k. `STOP` _____

l. `.END` _____

[6] 10. Match each of the addressing modes below with the addressing equation which defines it. (Two of the modes are not available on the Pep/7 virtual architecture, but were discussed in the textbook and lectures.) Write the letter preceding each addressing equation in the blank preceding the corresponding node name.

a. `Oprnd = Mem[SP + OprndSpec]`
b. `Oprnd = OprndSpec`
c. `Oprnd = Mem[Mem[OprndSpec]]`
d. `Oprnd = Mem[B + X]`
e. `Oprnd = Mem[Mem[SP + OprndSpec]]`
f. `Oprnd = Mem[OprndSpec]`

____ direct ____ indirect

____ immediate ____ stack-relative

____ indexed ____ stack-relative deferred

[4] 11. Suppose you are designing a compiler that represents the boolean value `false` with `0000` (hex) and `true` as any non-zero bit pattern. Given the following declarations

```
false:      .EQUATE  h#0000
true:       .EQUATE  h#0001   ; any non-zero value will do
p:          .BLOCK   d#2
q:          .BLOCK   d#2
```

match each code segment below with the Boolean operator that it implements. The object code leaves the value of the expression in the accumulator. The symbol `continue` labels the line of assembly code immediately following each excerpt shown.

a.
```
            LOADA    p,d
            BREQ     continue
            LOADA    q,d
```

b.
```
            LOADA    p,d
            BRNE     continue
            LOADA    q,d
```

c.
```
            LOADA    p,d
            BRNE     setFalse
            LOADA    true,i
            BR       continue
setFalse:   LOADA    false,i
```

d.
```
            LOADA    p,d
            BRNE     setFalse
            LOADA    q,d
            BRNE     setFalse
            LOADA    true,i
            BR       continue
setFalse:   LOADA    false,i
```

____ not p ____ p and q ____ p or q ____ p nor q

[5] 12. Given the following grammar

N = {A, B}
T = {0, 1}
P = the productions
1. A → 0B
2. B → 10B
3. B → ε
S = A

decide whether each of the following strings can be derived from the rules of the grammar or not. Circle "Yes" if the string is derivable and "No" if it is not.

Yes No 010

Yes No 10

Yes No 0110

Yes No 0

Yes No 010B

Answer the following questions on the separate pages of paper provided. Label each of your responses with the number of the corresponding question.

[5] 13. Draw a deterministic finite state machine (FSM) that recognizes strings of 1's and 0's of arbitrary length that contain the pattern 101 at least once anywhere in the string. For example, the FSM should accept 101, 101110, 01000101, and 00101011, but reject 0111, 100100, and 10.

[8] 14. (a) Draw the FSM that corresponds to the following state transition table, assuming that W is the start state and Z is the final state.

Current state	**Next state**			
	a	**b**	**c**	ε
W			X	X
X	Y	Z		
Y				X
Z				

(b) Transform the FSM from part (a) to the equivalent machine without the empty transitions.

(c) List four different strings which are accepted by this FSM.

[8] 15. Write an assembly language subroutine which implements the following C++ function:

```
void calcOvtm( int reg, int & ovrtm )
{
 ovrtm = reg + reg / 2;
}
```

Assume that the first parameter was pushed to the stack first and that the address of the second parameter was pushed to the stack second. Define the symbols `reg` and `ovrtm` so that the subroutine parameters can be accessed symbolically.

[4] 16. Draw the nonabbreviated logic diagram which corresponds to the following Boolean expression:

$((ab \oplus b')' + a'b)'$

[10] 17. Use a Karnaugh map to find the minimum OR of AND expression for each of the following:

a. $x(a, b, c) = \sum(0, 1, 2, 4, 6)$

b. $x(a, b, c) = \sum(1, 7) + d(3, 4)$

c. $x(a, b, c, d) = \sum(0, 2, 3, 4, 5, 6, 8, 10, 11, 13)$

[10] 18. a) Draw the circuit for a sequential net with one JK flip-flop, FFA; one T flip-flop, FFB; and one input, X, with flip-flop inputs

$J = X \oplus B$ $T = X \oplus A$

$K = \overline{X} B$

and output

$Z = A B$.

b) Show the state transition table for this circuit.

c) Draw the state transition diagram for this circuit.

[4] 19. Show how a *set-dominated flip-flop* can be implemented using a clocked SR flip-flop. A set-dominated flip-flop is like an SR flip-flop except that if both the S and R inputs are 1, the output Q is set to 1 (i.e., the set input S *dominates* the reset input R).