Section I: Multiple choice - 1 point each

1. What is the decimal equivalent of the IEEE-754 floating point number:
`1 10000000 10010000000000000000000`

a. `-17.0` **b.** `-8.5` **c.** `-4.25` **d.** `-2.125` **e.** `-1.0625` **f.** `-3.125`

2. Convert the decimal number **`#-476`** to a 12-bit two's complement binary number, and represent the result as hexadecimal

a. `x1BC` **b.** `x1BD` **c.** `x1DC` **d.** `xE23` **e.** `xE24` **f.** `xE44`

3. Given the instruction (located at address x3500)

```
        x3500    LDI R1, label1
```

and given:
label1 translates to address x35A0, which contains the value xC000;
memory location xC000 contains the value xD000;
memory location xD000 contains the value xFFFF;
What value will R1 contain after the instruction executes?

a. `x3500`
b. `xC000`
c. `35A0`
d. `xD000`
e. `xFFFF`

4. Simplify the Boolean expression:

`a.b'.c'.d + a.b'.c.d + a.b.c'.d + a.b.c.d' + a.b.c.d`

a. `a.b.c + a.b.d + b.c.d`
b. `a.d + a.b.c`
c. `a.c + a.b.d`
d. `a.c.d + a.b`
e. `a.c + b.d + b.c`
f. can't be simplified

5. How many memory locations can be addressed by a microprocessor that uses 24 bit addressing?

a. 16 k
b. 512 k
c. 16 M
d. 2 G
e. 4 G
f. none of the above

6. How many *select* lines does an 8 input multiplexer have?

a. 1 **b.** 2 **c.** 3 **d.** 8 **e.** 64 **f.** 256

7. How many *input* lines does an 8 input multiplexer have?

a. 1 **b.** 2 **c.** 3 **d.** 8 **e.** 64 **f.** 256

8. How many *output* lines does an 8 input multiplexer have?

a. 1 **b.** 2 **c.** 3 **d.** 8 **e.** 64 **f.** 256

9. How many *inputs* does a full adder circuit have?

a. 1 **b.** 2 **c.** 3 **d.** 4
e. It depends on the number of bits in the numbers being added

10. A "gate delay" can be described as the time needed for the output of a gate to settle to its correct level after one of its inputs has been changed. The full-adder circuit we have designed would therefore result in a gate delay of 2 units.
How many units of gate delay would a 4-bit ripple-adder display?

a. 2 **b.** 4 **c.** 8 **d.** 16 **e.** 32

11. How do we turn 8 separate gated D-latches into a single 8-bit register?
(WE = "Write Enable")

- **a.** connect the WE of each one separately to an output of a 3-bit decoder, and multiplex their outputs with the same decoder
- **b.** connect the inputs of each one separately to an output of a 3-bit decoder, and multiplex their WEs with the same decoder
- **c.** Use a 3-bit decoder as a selector, multiplex their WEs
- **d.** connect their WE lines together to a single external WE
- **e.** super-glue them together

12. How do we turn 8 separate gated D-latches into 8 addressable registers, each 1-bit wide?

- **a.** connect the WE of each one separately to an output of a 3-bit decoder, and multiplex their outputs with the same decoder
- **b.** connect the inputs of each one separately to an output of a 3-bit decoder, and multiplex their WEs with the same decoder
- **c.** Use a 3-bit decoder as a selector, multiplex their WEs
- **d.** connect their WE lines together to a single external WE
- **e.** super-glue them together

13. The following very complex subroutine moves the cursor down to the next line. What, if anything, is wrong with it?

```
            .ORIG   x4000
NEWLINE     LD R0,  CRLF
            OUT
            RET
CRLF        .FILL   x0A
```

- **a.** Nothing - it's fine as it is
- **b.** x4000 is not a valid starting address for a subroutine
- **c.** it should save and restore the contents of R0
- **d.** it should save and restore the contents of R7
- **e.** both c) and d)

14. A subroutine was called using a conditional branch (BR) instruction. The subroutine ends, as usual, with a RET instruction. What will happen when the subroutine terminates?

- **a.** Control will return to the original BR instruction
- **b.** Control will return to the instruction following the BR instruction
- **c.** Control will return to an unknown instruction, either crashing the program or producing unpredictable results.
- **d.** It will depend on which of the NZP condition codes the BR instruction tested

The next three questions refer to the following scenario:
Consider the controller of an elevator that connects the 1st, 2nd, 3rd & 4th floors of a building as an example of a **finite state machine**.
The states of this FSM correspond to the elevator stopped at each of the four floors, ***either*** with the doors closed, ***or*** with the doors open (i.e. "doors open" and "doors closed" are separate states).
Transitions between states will thus correspond to the elevator moving between floors, including “express” transitions between non-adjacent floors, and opening and closing doors while stopped at any given floor (obviously, the elevator can move only if the doors are closed)
External inputs are determined by the elevator call buttons.

15. How many separate states does this finite state machine have?

a. 4	**c.** 8	**e.** 16
b. 6	**d.** 12	**f.** 32

16. How many distinct transitions are there between states (***not*** including "null transitions", i.e. transitions from a state back to itself)?

a. 8	**c.** 16	**e.** 24
b. 12	**d.** 20	**f.** insufficient information

17. How many bits would the Storage Logic component of the fsm have to store?

a. 1	**c.** 3	**e.** 5
b. 2	**d.** 4	**f.** huh??

18. A computer system has a **word addressable** memory. Each word is 32 bits (4 bytes) wide. There are 24 memory address lines. What is the maximum available system memory?

a. 64k bytes	**c.** 64M bytes	**e.** 4G bytes
b. 16M bytes	**d.** 256M bytes	**f.** 16G bytes

19. How many data lines are required for the system in the previous question?

a. 8	**c.** 24	**e.** 28
b. 16	**d.** 26	**f.** 32

20. If the system in the previous questions were instead **byte-addressable**, and were to retain the same total number of bytes of memory, how many address lines would it require?

a. 8	**c.** 24	**e.** 28
b. 16	**d.** 26	**f.** 32

21. Given a very peculiar memory system that uses 22 bit-addressing, and is "three-bit" addressable (i.e. each location in memory stores 3 bits), how many ***bits*** of storage does the memory contain in total?

a. 88 bits	**c.** 1 Mbits	**e.** 16 Mbits
b. 64kbits	**d.** 12 Mbits	**f.** 64 Mbits

22. The central idea in the von Neumann model is that the program and data both reside in:

a. multiplexors	**c.** adders	**e.** switches
b. decoders	**d.** memory	**f.** models

The next three questions refer to the following system:
A certain ISA has a 32-bit word size, uses single word (32-bit) instructions, has 60 opcodes, 32 registers, and 4Gbyte of byte-addressable memory.
One group of instructions in this ISA takes the form:
```
OPCODE | DESTINATION REGISTER | SOURCE REG. | Flag | IMMEDIATE VALUE
```
Or
```
OPCODE | DESTINATION REGISTER | SOURCE REG. 1 | Flag | SOURCE REG. 2
```
A single bit in the instruction ("Flag") is used to differentiate these two addressing modes.
Another group of instructions takes the form
```
OPCODE | SOURCE/DESTINATION REGISTER | PC OFFSET
```
Where PC Offset is the 2's complement "distance" from the current PC to the labelled location.

23. What is the range of values that can be stored in the "Immediate" field (as a 2's complement value)?

a. -16 to +15
b. 0 to (64k – 1)
c. –16k to +(16k – 1)
d. –32k to +(32k – 1)
e. –64k to +(64k – 1)

24. How many bits are required for addressing (i.e. what is the size in bits of an address)?

a. 16
b. 24
c. 32
d. 48
e. 4G

25. How far (in memory locations) can the label be from an instruction that references the label, using the <opcode | register | pc offset> addressing mode?
You may assume that in this assembly language, each line of source code assembles to a single one-word (32-bit) instruction.

a. + / - 32k
b. + / - 64k
c. + / - 128k
d. + / - 256k
e. + / - 512k
f. + / - 1M

26. An ISA specifies a *word size* of 8 bytes; *word addressability*; and an *address space* of 4G; it uses single-word instructions (i.e. each instruction is a single 8 byte word).
What are the sizes of the PC and the IR?

a. both 24 bits
b. both 32 bits
c. both 64 bits
d. PC: 24 bits; IR: 64 bits
e. PC: 64 bits; IR: 32 bits
f. PC: 32 bits; IR: 64 bits

27. For the system in the previous question: What are the sizes of the MDR and the MAR?

a. both 24 bits
b. both 32 bits
c. both 64 bits
d. MAR: 32 bits; MDR: 64 bits
e. MAR: 64 bits; MDR: 32 bits
f. MAR: 24 bits; MDR: 64 bits

28. Consider the following LC-3 code fragment (the hex values in the first column give the address to which the corresponding instruction is loaded):

```
x34FE   loop1   ADD  R4, R5, R6
                 .....
x35A0    BRnp loop1   ; NOTE THE TEST IS "np"
```

Given: the BR opcode is `0000 NZP`
What does the instruction at x35A0 assemble to? *(Remember to account for the current value of the PC when the BR instruction is being executed, and the direction of the "jump", and the condition codes to be tested)*

a. x0B5D
b. x0B5E
c. x0B5F
d. x0AA1
e. x0AA2
f. x0AA3

29. How many separate decoders are there for the general purpose register bank in the LC-3?

a. 1 **b.** 2 **c.** 3 **d.** 4 **e.** 8
f. The register bank does not require decoders

30. In the ***Indirect mode*** of memory addressing in the LC-3 (e.g. the instructions LDI and STI), the Effective Address is calculated by:

a. `(BaseReg) + SEXT(IR[5:0])`
b. `(PC) + SEXT(IR[8:0])`
c. `Mem[(PC) + SEXT(IR[5:0])]`
d. `Mem[(PC) + SEXT(IR[8:0])]`
e. `(IR) + SEXT(PC[5:0])`
f. `Mem[(IR) + SEXT(PC[8:0])]`

31. What are the "micro-instructions" that comprise the Fetch phase of the Instruction cycle?

a. `IR <- (PC); MAR <- (IR); PC <- Mem[MDR]; PC <- (PC) + 1;`
b. `PC <- (MDR) + 1; MAR <- Mem[PC]; IR <- (MDR)`
c. `MAR <- (PC) + 1; MDR <- Mem[IR]; IR <- (MAR);`
d. `MAR <- (PC); PC <- (PC) + 1; MDR <- Mem[MAR]; IR <- (MDR)`
e. `MDR <- (PC); PC <- (PC) + 1; MAR <- Mem[MDR]; IR <- (PC)`

32. How does the control unit decide whether to take the branch pointed to in a BR instruction? (n, z & p represent IR [11:9]; N, Z & P represent the values of the condition code registers)

a. if `(n . N + z . Z + p . P) = 1`, the MAR is write enabled
b. if `(n . N + z . Z + p . P) = 1`, the PC is write enabled
c. if `(n . N + z . Z + p . P) = 1`, the EA corresponding to the label is calculated
d. if `( (n + N) . (z + Z) . (p + P) ) = 1`, the MAR is write enabled
e. if `( (n + N) . (z + Z) . (p + P) ) = 1`, the PC is write enabled
f. if `( (n + N) . (z + Z) . (p + P) ) = 1`, the EA corresponding to the label is calculated

33. All ***control*** instructions in the LC-3 have one main step in common:

a. They all reconstruct the required memory address in the same way
b. They all use the ALU in reconstructing the required memory address
c. They all write to the IR in the execution phase of the instruction cycle
d. They all write to the PC in the execution phase of the instruction cycle
e. They all write to the MDR in the execution phase of the instruction cycle
f. They all write to the GPR bank in the execution phase of the instruction cycle

34. The LC-3 instruction cycle consists of 6 phases. What does this mean?

a. 6 instructions require 1 cycle, while the other instructions may require more
b. Each instruction consists of up to 6 steps
c. The execute stage has 6 possible operations
d. 6 of the 16 bits of an instruction are dedicated to opcode
e. The processor has 6 main parts, including the ALU, register file, etc.

35. One of the four control signals to the LC-3 ALU is "pass-through input A" - i.e. input A is connected directly to the output.
Which of the following instructions would use this control signal?

a. NOT
b. LD, LDI & LDI
c. ST, STI & STR
d. BR & JMP
e. JSR/JSRR
f. TRAP

36. In the LC-3, the DR decoder input comes from the DRMUX. What are two of the inputs to the DRMUX? *(Hint: think about the JSR and TRAP instructions)*

a. `IR[2:0], IR[8:6] and IR[11:9]`
b. `IR[8:6] and IR[11:9]`
c. `[111] and IR[11:9]`
d. `The system bus and IR[11:9]`
e. `(PC) and IR[11:9]`
f. `PC[2:0] and IR[11:9]`

37. In the LC-3, the SR1 decoder input comes from the SR1MUX. What are two of the inputs to the SR1MUX? *(Hint: think about the load and store instructions)*

- **a.** `IR[2:0], IR[8:6] and IR[11:9]`
- **b.** `IR[8:6] and IR[11:9]`
- **c.** `[111] and IR[11:9]`
- **d.** `The system bus and IR[11:9]`
- **e.** `(PC) and IR[11:9]`
- **f.** `PC[2:0] and IR[11:9]`

38. In the LC-3 (and most ISAs), the System Control Block, or Trap Vector Table, contains:

- **a.** the complete Trap Service Routines
- **b.** the 9-bit PC offsets of the Trap Service Routine addresses
- **c.** the 8-bit entry point into the Trap Vector Table
- **d.** the starting addresses of the Trap Service Routines
- **e.** the return addresses to be used after returning from a Trap Service Routine

39. What is the main purpose of the first pass of a two-pass assembler?

- **a.** to determine if the code will fit into available memory
- **b.** to produce the machine language equivalent of the assembly language instructions
- **c.** to link other possible object files in order to create the executable
- **d.** to remove all pseudo-ops from the code before it is assembled
- **e.** to build a symbol table relating labels to memory addresses

40. In a cpu that uses the technique of memory mapping to address ports (registers that interface between the cpu and peripherals), how ***must*** the ports actually be accessed?

- **a.** via dedicated i/o instructions.
- **b.** via interrupts
- **c.** via polling
- **d.** via standard Load/Store instructions
- **e.** either a) or d)
- **f.** none of the above

41. What component of the cpu gets "interrupted" by an Interrupt signal? (i.e. where does the Interrupt signal go to?)

- **a.** The PC
- **b.** The PC-MUX
- **c.** The MAR-MUX
- **d.** The global bus
- **e.** The FSM
- **f.** The Memory Mapping logic

42. In Interrupt processing, what is the purpose of the IACK signal?

- **a.** It alerts a collection of peripherals that the cpu is available and will service interrupts
- **b.** It polls multiple peripherals to find which initiated an interrupt
- **c.** It is used by the cpu to flag to a peripheral that it has completed servicing its interrupt
- **d.** It is used by a peripheral to flag to to the cpu that it no longer requires servicing
- **e.** It is used by a peripheral to "lock" the bus once its interrupt has been acknowledged by the cpu

43. In the LC-3, the TRAP instruction and the interrupt handler both manage the invocation of service routines in a similar fashion. Specifically, both use:

- **a.** polling of status registers to decide when to read from/write to a port
- **b.** a trap/interrupt vector as an entry point into a table of service routine addresses
- **c.** an IACK signal to determine which service routine is requested
- **d.** a stack to store information required for the return, allowing nested calls
- **e.** a system of task priority comparisons to determine whether to invoke the service routine

44. What system state information has to be saved before an interrupt-enabled LC-3 can proceed with servicing an interrupt?

- **a.** the value of every control signal produced by the finite state machine
- **b.** the value of every control signal produced by the finite state machine, plus the contents of all Registers (GPRs, PC, condition codes, etc.) – except the IR
- **c.** the PC
- **d.** the PC and all the General Purpose Registers
- **e.** the PC, and the PSR (Processor Status Register, containing the NZP condition codes, the Privilege level, and the current task priority)
- **f.** the PC and the MCR (Machine Control Register)

45. At what point in the instruction cycle is an interrupt handled?

- **a.** at any time during the entire cycle - the interrupt is just one of several external inputs to the FSM
- **b.** at any time during the fetch instruction phase
- **c.** only at the very start of the fetch instruction phase
- **d.** at any time during the last phase (store)

46. The data protocol of a stack data structure is known as

- **a.** LIFO (Last In, First Out)
- **b.** FIFO (First In, First Out)

47. The two main approaches to converting a HLL source code to ML (Machine Language) are:

- **a.** direct and indirect
- **b.** memory mapping and polling
- **c.** assembly and disassembly
- **d.** interpreting and compiling
- **e.** compiling and linking

48. The structure which allows Higher Level Languages to make nested function calls is:

- **a.** Activation records stored on the run-time stack
- **b.** The symbol table
- **c.** The frame pointer
- **d.** The Processor Status Register
- **e.** The Machine Control Register

49. Which LC-3 assembly language instruction is most likely to be used to compile a HLL (Higher Level Language) access to a local variable:

a. LDR **b.** LDI **c.** LD **d.** TRAP **e.** JSR

50. Which register is most likely to be used as the Base Register in accessing the variable, in the situation described in the previous question?

- **a.** Frame pointer (R5 in the LC3)
- **b.** Top of Stack (R6 in the LC3)
- **c.** Program Counter (R7 in the LC3)
- **d.** Processor Status Register

Last name: ____________________ First name: __________________________

<u>Section II: Written answers</u>

1. **<u>15 points</u>**
   Design a digital combinational logic circuit with four inputs: a, b, c & d, where (a, b) represents one 2-bit unsigned binary number A{1:0]; and (c, d) represents another 2-bit unsigned binary number B[1:0] (i.e. both A and B are in the range 0 to 3).
   The circuit has 4 outputs (or you can regard it as being 4 distinct circuits, each with a single bit output) – in other words, the truth table will have 4 input columns and 4 output columns.
   These output columns together repesent the 4-bit ***product*** Y[3:0]
   Y = A * B
   For instance, inputs corresponding to "3 , 2" would output bits corresponding to 6

   - Start by drawing up the truth table (6 points) (show ***only*** those rows which produce a 1 in any of the poutput columns)
     Make sure you label your input and output columns correctly – everything else depends on getting the table right!
   - then derive the algebraic expression for the third bit of the output, Y[2] (3 points);
   - and simplify it (3 points)
   - Finally, draw the resulting circuit (3 points)

   (Each part is "all or nothing" - no partial credit)

Last name: ______________________ **First name:** ____________________________

Last name: ____________________ **First name: __________________________**

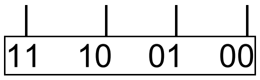
2. **15 points**

Given the data path of the LC-3 (provided below), give a complete description of the STORE DIRECT instruction (`ST SR, label`), as follows:

a) Give the RT (Register Transfer) specification of the instruction (4 points)

b) List the data applied to all relevant circuits in the data path (i.e. just those circuits relevant to the ST instruction); and list, in the correct sequence, every control signal set by the FSM to implement this instruction. (11 points)

Remember that not every tri-state device may be actually depicted on the schematic – but you must still specify the control signal if it is involved (make up descriptive names for any control signals that are not shown on the schematic). Likewise, the SR MUX is not shown, but you must still include the correct select control signal and data input for it.
For multi-bit control signals, you must specify the actual value where possible:
e.g. if the control signal is a 2-bit MUX selector, selecting for the input that is third from the right, then you must specify the value 10



If several control signals act at the same time, indicate that fact; otherwise list them in sequence.

Last name: ______________________ **First name: ____________________________**

Last name: ____________________ First name: __________________________

3. **<u>15 points</u>**
Construct the Finite State Machine representation for a counter with a cycle length of 4 - i.e. a circuit that counts 0 – 1 – 2 – 3 (output as a binary value, obviously) with successive clock pulses, and then starts over.
The external output is the 2-bit count.
The only external input is R, a reset pulse: when R = 1 it resets the next count to 0, no matter what the current state; when R = 0 it keeps counting (i.e. the system transitions to the next state in sequence).

Then construct the complete truth table(s) for the device, showing
the inputs: "current state" labels (ie. the state we are transitiong from), and R
the outputs: "next state" labels (i.e. the state we are transitioning to), and the 2-bit count associated with that state.
(Hint: if you choose the state labels sensibly, they will be identical to the output)

Finally, derive and simplify the algebraic expression for bit 0 of the output.

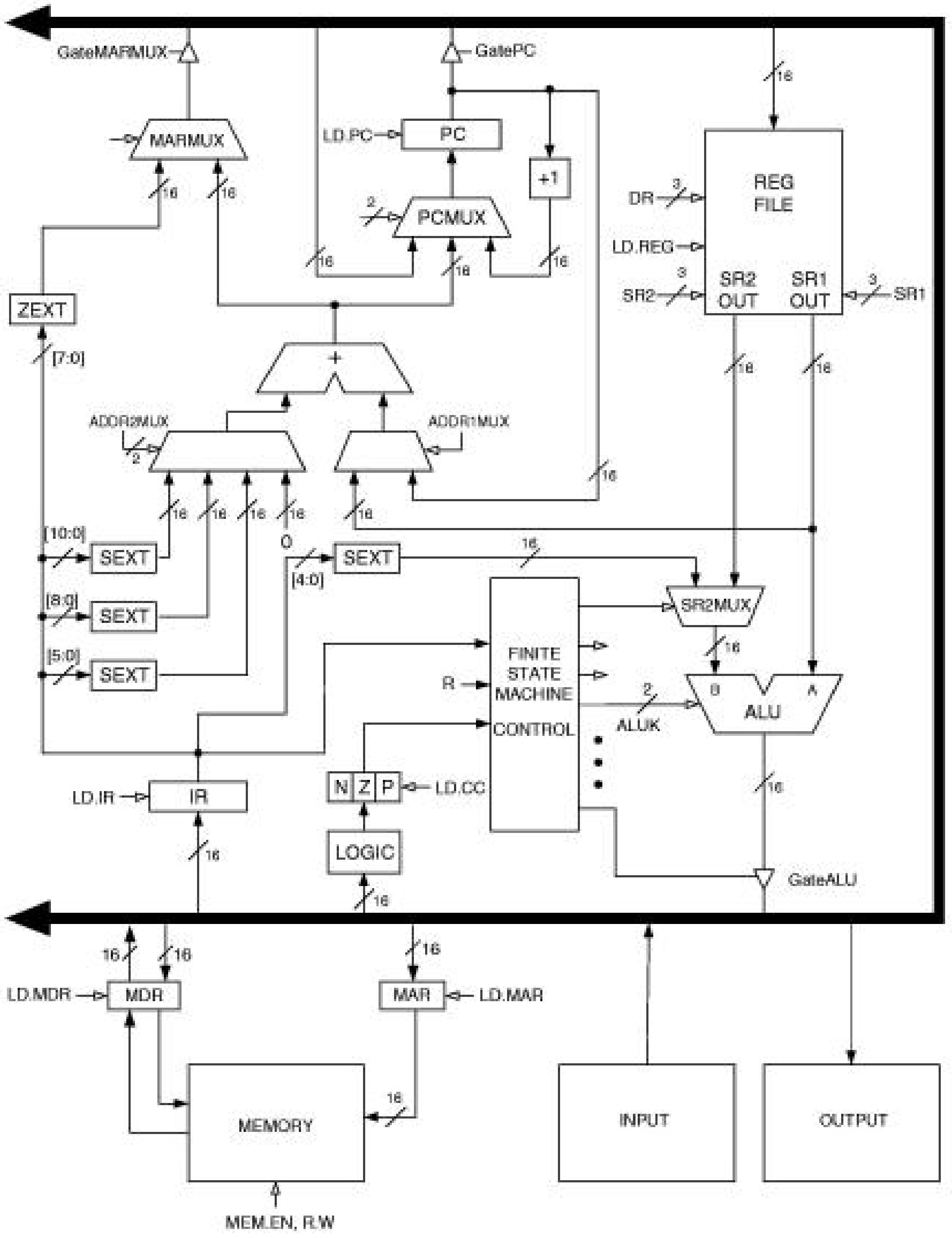
Last name: ______________________ **First name: ____________________________**

Last name: ______________________ First name: ____________________________

4. **5 points**
   A number of LC-3 instructions have an "evaluate address" step in the instruction cycle, in which a 16-bit address is constructed and written to the Memory Address Register via the MARMUX.

   List ***all*** LC-3 instructions that write to the MAR during the evaluate address phase of the instruction cycle, with the Register Transfer description of each.

Last name: _______________________ **First name:** ______________________________


GateMARMUX
GatePC
MARMUX
LD.PC
PC
+1
PCMUX
REG FILE
DR
LD.REG
SR2
SR2 OUT
SR1 OUT
SR1
ZEXT
[7:0]
ADDR2MUX
ADDR1MUX
[10:0]
SEXT
[8:0]
[5:0]
[4:0]
0
SR2MUX
FINITE STATE MACHINE
R
CONTROL
ALUK
ALU
LD.IR
IR
N Z P
LD.CC
LOGIC
GateALU
LD.MDR
MDR
MAR
LD.MAR
MEMORY
INPUT
OUTPUT
MEM.EN, R.W
16