Department of Electrical and Computer Engineering

The University of Texas at Austin
EE 460N, Fall 2020
Problem Set 5
Yale N. Patt, Instructor
Chester Cai, Sean Stephens, Arjun Ramesh TAs

This problem set is not to be graded. However, anything covered on this problem set is fair game for the final exam.

Instructions

You are encouraged to work on the problem set in groups and turn in one problem set for the entire group. The problem sets are to be submitted on Gradescope. Only one student should submit the problem set on behalf of the group. Please see the email regarding Gradescope submission for details. The entry code is MYRDPJ.

You will need to refer to the assembly language handout and the LC-3b ISA, microarchitecture, and state diagram documents on the course website.

Questions

Problem 1

Determine the decimal value of the following IEEE floating point numbers.

1. `1 10000000 10100000000000000000000`
2. `0 00000000 01010000000000000000000`
3. `1 11111111 00000000000000000000000`

Problem 2

From Tanenbaum, 4th edition, Appendix B, 4.

The following binary floating-point number consists of a sign bit, an excess 63, radix 2 exponent, and a 16-bit fraction. Express the value of this number as a decimal number.

```
0 0111111 0000001111111111
```

Problem 3

From Tanenbaum, 4th edition, Appendix B, 5.

To add two floating point numbers, you must adjust the exponents (by shifting the fraction) to make them the same. Then you can add the fractions and normalize the result, if need be. Add the single precision IEEE floating-point numbers 3EE00000H and 3D800000H and express the normalized result in hexadecimal. ['H' is a notation indicating these numbers are in hexadecimal]

Problem 4

From Tanenbaum, 4th edition, Appendix B, 6.

The Tightwad Computer Company has decided to come out with a machine having 16-bit floating-point numbers. The model 0.001 has a floating-point format with a sign bit, 7-bit, excess 63 exponent and 8-bit fraction. Model 0.002 has a sign bit, 5-bit, excess 15 exponent and a 10-bit fraction. Both use radix 2 exponentiation. What are the smallest and largest positive normalized numbers on both models? About how many decimal digits of precision does each have? Would you buy either one?

Problem 5

The following numbers are represented exactly with a 9-bit floating point representation, in the format of the IEEE Floating Point standard:

-infinity, -1, 0, 5/16, 19.5, 48.

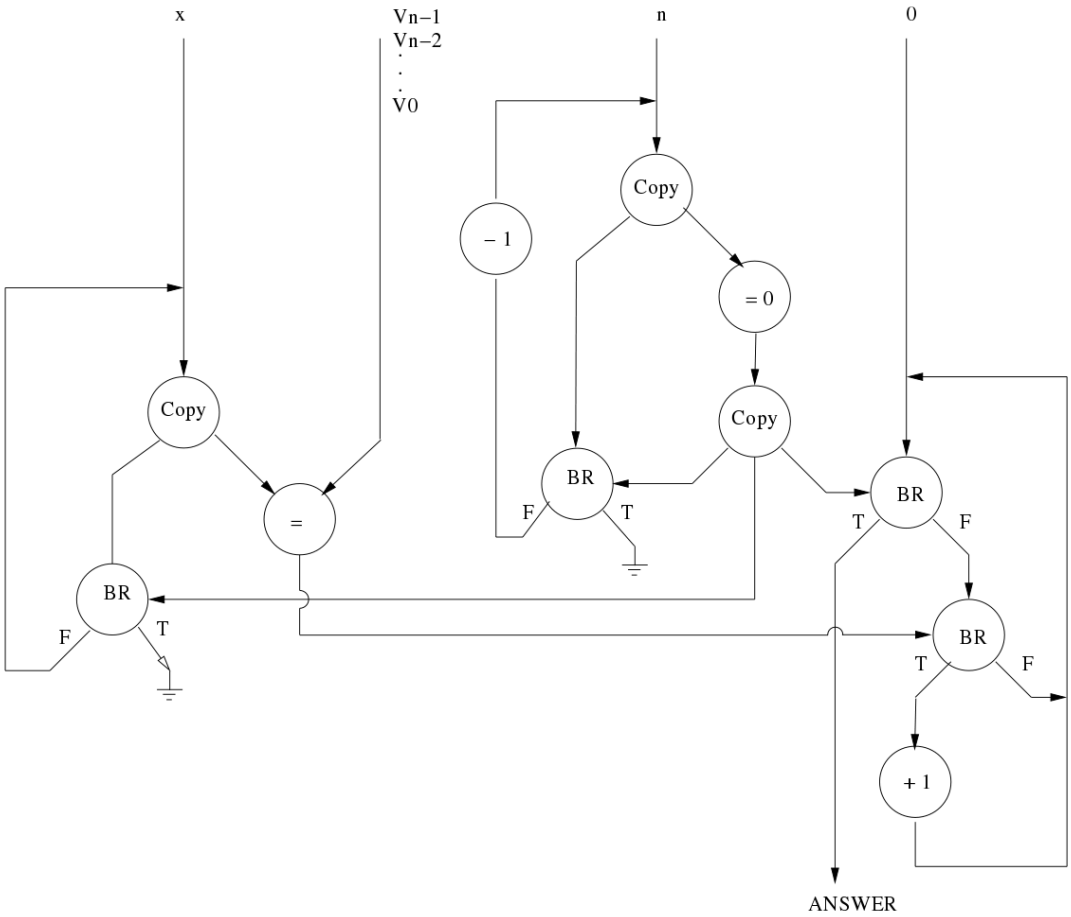
1. How many bits are needed for the fraction?
2. What is the bias?
3. Write each number in in the 9-bit floating point representation, below:

Value	Representation
48	
19.5	

5/16	
0	
-1	
-infinity	

Problem 6

The following data flow graph receives as inputs a value x, an n element vector V0, V1, ..., Vn-1, the value n, and a value 0 on its four input ports.



What "answer" is produced by the execution of this data flow graph?

Problem 7

We must compute the following expression:

```
a*x^6 + b*x^5 + c*x^4 + d*x^3 + e*x^2 + f*x + g
```

a. How many operations and time-steps will the computation take on a single processor system (Use the smallest number of operations possible)?
b. How many operations and time-steps will the computation take on a multiprocessor system with 4 processors? (Use the smallest number of operations possible)
c. What is the speedup of the multiprocessor system over a single processor?

Problem 8

Speed-up with p processors is defined as T1/Tp, where T1 is the time to solve the problem with one processor and Tp is the time to solve the problem if you have p processors. What important requirement is there on T1?

~~**Problem 9**~~

~~In an Omega network as presented in class, assume that there are n inputs and n outputs. Let k be the size of each switch. For k taking the values 2, 4, 8, and 64, answer the following questions. (Assume the cost of each switch is k^2)~~

~~a. What is the cost of the network as a function of n?~~
~~b. What is the latency of the network?~~
~~c. Assume that n=64. What k value would you choose? Why? State your assumptions and design point.~~

Problem 10

Consider the following piece of code:

```
for(i = 0; i < 100; i++)
  A[i] = ((B[i] * C[i]) + D[i]) / 2;
```

a. Translate this code into assembly language using the following instructions in the ISA (note the number of cycles each instruction takes is shown with each instruction):

Opcode	Operands	Number of Cycles	Description
LEA	Ri, X	1	Ri ← address of X
LD	Ri, Rj, Rk	11	Ri ← MEM[Rj + Rk]
ST	Ri, Rj, Rk	11	MEM[Rj + Rk] ← Ri
MOVI	Ri, Imm	1	Ri ← Imm
MUL	Ri, Rj, Rk	6	Ri ← Rj × Rk
ADD	Ri, Rj, Rk	4	Ri ← Rj + Rk
ADD	Ri, Rj, Imm	4	Ri ← Rj + Imm
RSHFA	Ri, Rj, amount	1	Ri ← RSHFA (Rj, amount)
BRcc	X	1	Branch to X based on condition codes

Assume it takes one memory location to store each element of the array. Also assume that there are 8 registers (R0-R7).

How many cycles does it take to execute the program?

b. Now write Cray-like vector/assembly code to perform this operation in the shortest time possible. Assume that there are 8 vector registers and the length of each vector register is 64. Use the following instructions in the vector ISA:

Opcode	Operands	Number of Cycles	Description
LD	Vst, #n	1	Vst ← n
LD	Vln, #n	1	Vln ← n
VLD	Vi, X	11, pipelined	
VST	Vi, X	11, pipelined	
Vmul	Vi, Vj, Vk	6, pipelined	
Vadd	Vi, Vj, Vk	4, pipelined	
Vrshfa	Vi, Vj, amount	1	
Vbrcc	X	1	

c. How many cycles does it take to execute the program on the following processors? Assume that memory is 16-way interleaved.

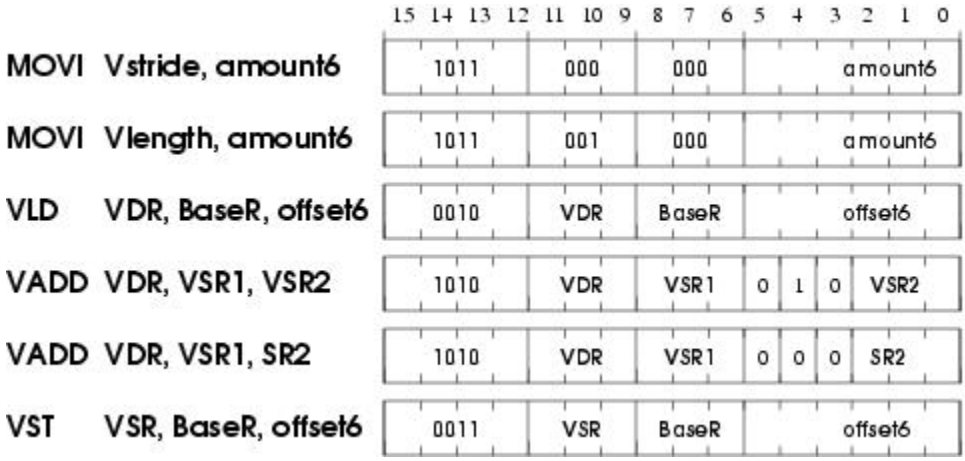
Vector processor without chaining, 1 port to memory (1 load or store per cycle)

Vector processor with chaining, 1 port to memory

Vector processor with chaining, 2 read ports and 1 write port to memory

Problem 11

Little Computer Inc. is now planning to build a new computer that is more suited for scientific applications. LC-3b can be modified for such applications by replacing the data type Byte with Vector. The new computer will be called LmmVC-3 (Little 'mickey mouse' Vector Computer 3). Your job is to help us implement the datapath for LmmVC-3. LmmVC-3 ISA will support all the scalar operations that LC-3b currently supports except the LDB and STB will be replaced with VLD and VST respectively. Our datapath will need to support the following new instructions:



Note: VDR means “Vector Destination Register” and VSR means “Vector Source Register.”

MOVI

If IR[11:9] = 000, MOVI moves the unsigned quantity amount6 to Vector Stride Register (Vstride). If IR[11:9] = 001, MOVI moves the unsigned quantity amount6 to Vector Length Register (Vlength). This instruction has already been implemented for you.

VLD

VLD loads a vector of length Vlength from memory into VDR. VLD uses the opcode previously used by LDB. The starting address of the vector is computed by adding the LSHF1(SEXT(offset6)) to BaseR. Subsequent addresses are obtained by adding LSHF1(ZEXT(Vstride)) to the address of the preceding vector element.

VST

VST writes the contents of VSR into memory. VST uses the opcode previously used by STB. Address calculation is done in the same way as for VLD.

VADD

If IR[4] is a 1, VADD adds two vector registers (VSR1 and VSR2) and stores the result in VDR. If IR[4] is a 0, VADD adds a scalar register (SR2) to every element of VSR and stores the result in VDR.

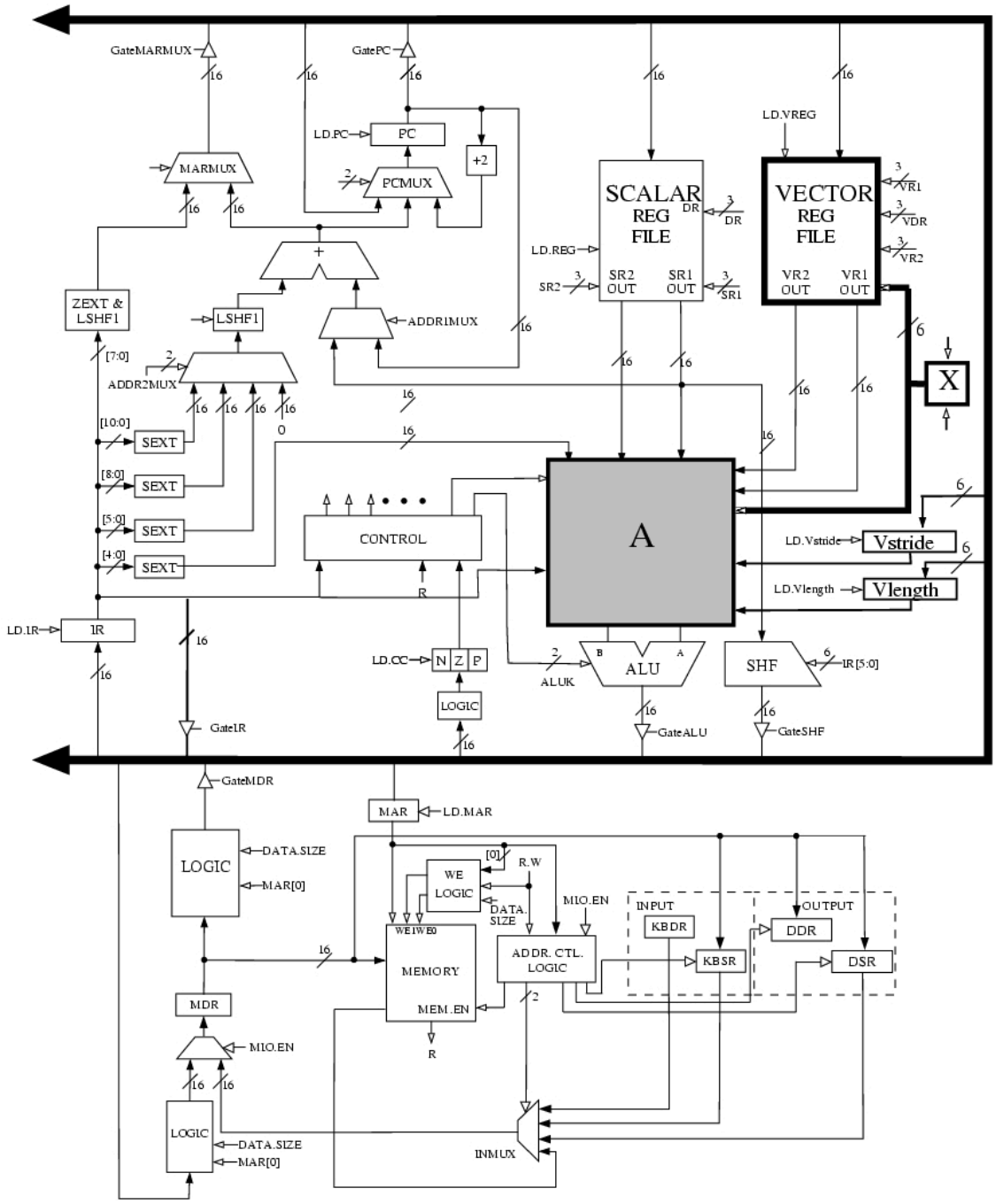
VLD, VST, and VADD do not modify the content of Vstride and Vlength registers.

The following five hardware structures have been added to LC-3b in order to implement LmmVC-3.

- Vector Register File with eight 63-element Vector registers
- Vector Length Register

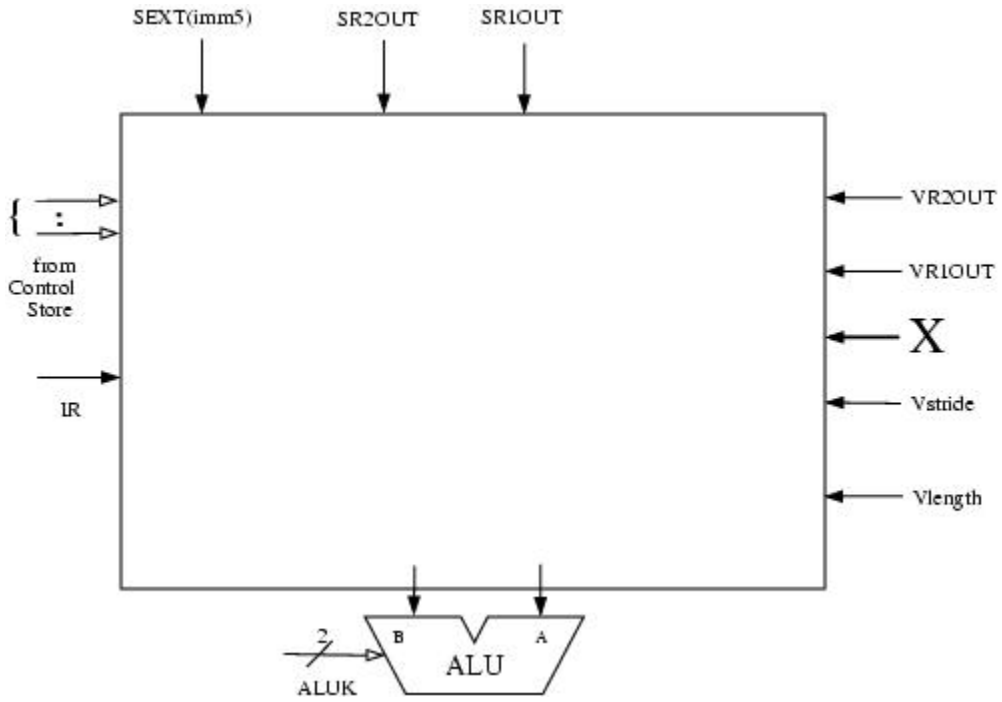
- Vector Stride Register
- A third input to DRMUX containing IR[8:6]
- Grey box A
- Box labeled X

These structures are shown in the LmmVC-3 datapath diagram:

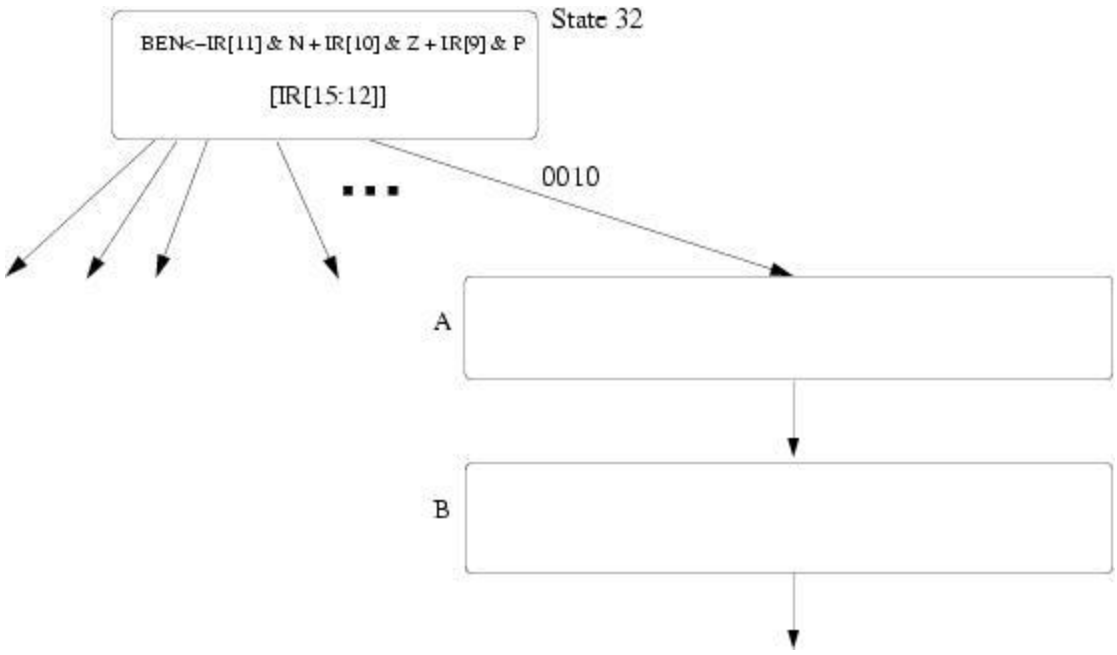


a. A 6-bit input to the Vector Register file has been labeled X on the datapath diagram. What is the purpose of this input? (Answer in less than 10 words)

b. The logic structure X contains a 6-bit register and some additional logic. X has two control signals as its inputs. What are these signals used for?
c. Grey box A contains several additional muxes on both input lines to the ALU. Complete the logic diagram of grey box A (shown below) by showing all muxes and interconnects. You will need to add new signals to the control store; be sure to clearly label them in the logic diagram.
    - Keep in mind that we will still need to support all the existing scalar operations.
    - The XOR operation in the ALU can be used to compare two values.
    - Our solution required 3 additional control signals and 6 2-to-1 muxes.



d. We show the beginning of the state diagram necessary to implement VLD. Using the notation of the LC-3b State Diagram, add the states you need to implement VLD. Inside each state describe what happens in that state. You can assume that you are allowed to make any changes to the microsequencer that you find necessary. You do not have to make/show these changes. You can modify BaseR and the condition codes. Make sure your design works when Vlength equals 0. Full credit will be awarded to solutions that require no more than 7 states.



Problem 12

A four processor system, each processor having its own cache , uses the Directory scheme to maintain cache coherence. The directory stores a bit vector for each "line" (or, "block") of memory, indicating its status relative to the caches. Assume no cache has line A. Then in sequence: processor 1 wishes to read a value in line A, processor 2 wishes to write a value in line A, processor 3 wishes to read a value in line A. At the end of this sequence, what are the contents of the bit vector for line A.

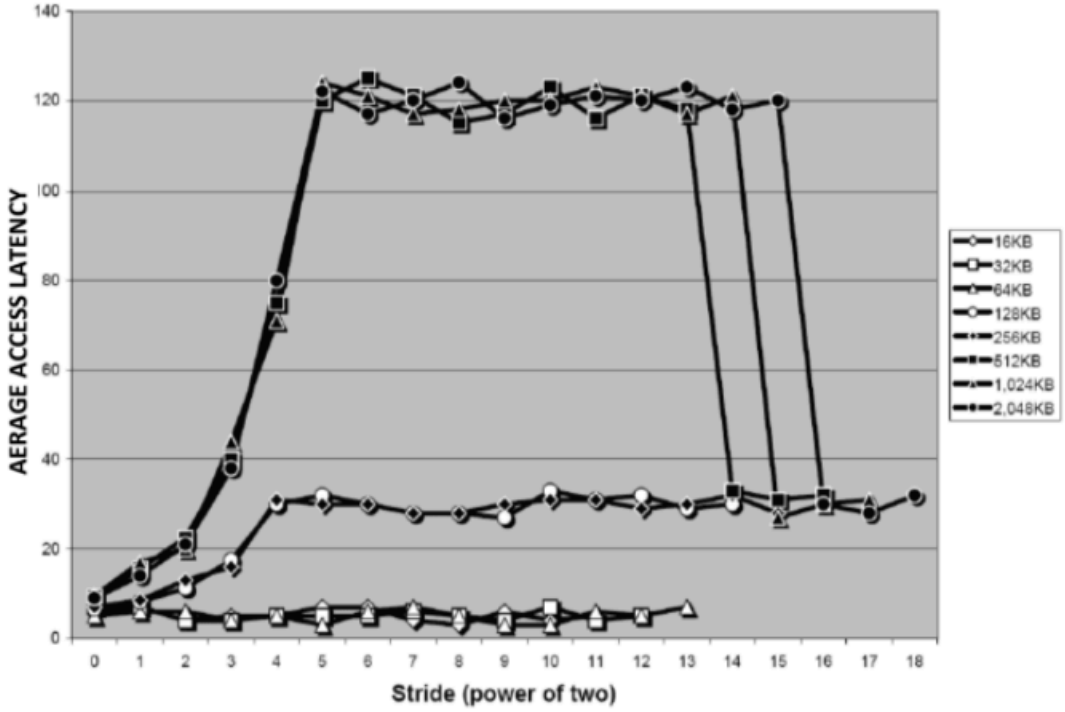
Problem 13

2. Cache operation.

The following program is used to derive the properties of a 2-level cache hierarchy. The program is run multiple times with different strides and array size parameters. The different results, shown in the graph below, reveal information about the cache parameters.

```
for(k = 0; k < LARGE_NUMBER; k++)
   for(i = 0; i < ARRAY_SIZE; i += Stride)
      s = s + a[i];
```

All variables are 4-byte integers. The figure below shows the average access latency for loads in the program as the array size and stride are varied. The y-axis shows the average load access latency and each point on the x-axis is for a different stride. Each line in the graph is for one particular array size : 16KB (4K entries), 32KB (8K entries), ...



Answer the following 8 questions and briefly justify each answer (very briefly). State assumptions in the box at the end. Assume that L2 access starts after an L1 miss (no concurrent access to L1 and L2 or L2 and memory). Also, **DO NOT** try to get exact values. Round off values to the nearest integer possible.

Hint: approach this question by "simulating" the first few cases with a small set associative cache to get a feel for the behavior; think of the different type of locality available and how locality impacts misses; you can work out the cache parameters separately; remember that powers of two are meaningful.

(a) (7 points) What is the hit time for L1 cache?

(b) (7 points) What is the hit time for L2 cache?

(c) (7 points) What is the miss penalty for L1 cache?

(d) (7 points) What is the miss penalty for L2 cache?

(e) (7 points) What is the block size for L1 cache?

(f) (7 points) What is the block size for L2 cache?

(g) (10 points) What is the associativity for L1 cache?

(h) (10 points) What is the associativity for L2 cache?